Deutsch

FUJITSU Software BS2000

SESAM/SQL-Server V9.0

Basishandbuch

Benutzerhandbuch

Ausgabe Oktober 2016

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2008

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2008 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Inhalt

1 Einleitung

Das Datenbanksystem SESAM/SQL-Server erfüllt durch seine Funktionen und seine Architekturmerkmale alle Anforderungen, die heute an einen leistungsfähigen Datenbankserver gestellt werden. Diese Eigenschaft drückt sich auch im Produktnamen SESAM/SQL-Server aus.

SESAM/SQL-Server gibt es als Standard Edition mit Singletask-Betrieb und als Enterprise Edition, die den Multitask-Betrieb beinhaltet.

Der Einfachheit halber ist im Folgenden von SESAM/SQL die Rede, wenn das Datenbanksystem SESAM/SQL-Server gemeint ist.

Kurzbeschreibung des Produkts

SESAM/SQL ist der relationale Datenbankserver auf BS2000-Systemen.

SESAM/SQL verbindet die Vorzüge des relationalen Datenmodells mit allen Eigenschaften, die für den Produktiveinsatz unter hoher Belastung erwartet werden. Dies bedeutet einerseits einfache Handhabung und Unabhängigkeit der Daten von der physikalischen Abspeicherung, andererseits die Eignung für hohe Transaktionsraten und große Datenmengen sowie ausgeprägte Sicherheits- und Verfügbarkeitseigenschaften.

Die SQL-Schnittstelle von SESAM/SQL ist konsequent an der aktuellen SQL-Norm ausgerichtet. Durch die standardisierte SQL-Schnittstelle lassen sich mit SESAM/SQL portable, zukunftssichere Datenbank-Anwendungen erstellen, die sich auf unterschiedliche Datenbank- und Betriebssysteme übertragen lassen.

SESAM/SQL erfüllt alle Anforderungen, die heute an ein modernes Datenbanksystem gestellt werden:

- SESAM/SQL verwendet mit SQL eine einheitliche Sprache und ein durchgängiges Begriffssystem für Definition, Aufbau, Wartung und Pflege einer relationalen Datenbank sowie für das Erstellen von Anwenderprogrammen.
- SESAM/SQL läuft auf allen BS2000-Systemen und bietet sich als leistungsfähiger SQL-Server für Clients an, die auf BS2000, UNIX-Systemen, Solaris, Linux und Windows-Systemen ablaufen.

- SESAM/SQL erlaubt die Verwendung von Unicode-Zeichen in Tabellen und berücksichtigt codierte Zeichensätze.
- SESAM/SQL zeichnet sich durch sehr hohe Verfügbarkeit, Sicherheit und Datenintegrität aus.
- SESAM/SQL verfügt über moderne Techniken zur Parallelverarbeitung im Mehrbenutzer- und Mehrdatenbankbetrieb.
- Der universelle Transaktionsmonitor openUTM und das Datenbanksystem SESAM/SQL bilden ein leistungsfähiges DB/DC-System mit voll koordinierter Transaktionsverarbeitung und Wiederanlauffähigkeit für Online-Anwendungen.
- Das Produkt SESAM/SQL-DCN ermöglicht den transparenten, effizienten und gesicherten Zugriff auf verteilte Datenbanken in BS2000-Rechnernetzen.
- Mit der in SESAM/SQL enthaltenen Release Unit SESAM-DBAccess, Kurzname SESDBA, kann auf SESAM/SQL-Datenbanken im Client-Server-Umfeld über folgende Schnittstellen zugegriffen werden:
  - aus Java-Programmen bzw. Java Server Pages, Java Servlets und Java Applets auf beliebigen Plattformen
  - über die von Microsoft definierte ADO-Technologie (ActiveX Data Objects) und der ADO.NET-Schnittstelle in Client-Anwendungen auf Windows-Systemen
  - über den PDO-Treiber von SESAM/SQL für die PHP-Schnittstelle (**P**HP: **H**ypertext **P**rocessor) auf einem Apache Webserver
- Eine Vielzahl von Zusatzprodukten erweitert die Einsatzmöglichkeiten von SESAM/SQL. Dies fängt an bei Werkzeugen für das Datenbankdesign, Programmiersprachen und Software-Entwicklungsumgebungen der dritten und vierten Generation und reicht hin bis zu einfach handhabbaren Produkten für den Datenbank-Endbenutzer und dem Einsatz von SESAM/SQL in Anwendungen für das World Wide Web.

1.1 Konzept der SESAM/SQL-Server-Dokumentation

Das Datenbanksystem SESAM/SQL-Server ist in folgenden Handbüchern dokumentiert:

- Basishandbuch
  Das Handbuch gibt einen Überblick über das Datenbanksystem und beschreibt Grundlagen, Konzepte, Zusammenhänge und Fachwörter. Es ist die Basis für das Verständnis der weiteren SESAM/SQL-Handbücher.
- SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen
  Das Handbuch behandelt die Programmeinbettung und beschreibt die SQL-Sprachelemente bzgl. Syntax und Semantik in alphabetischer Reihenfolge.
- SQL-Sprachbeschreibung Teil 2: Utilities
  Die Utility-Anweisungen sind in der alphabetischen Ordnung getrennt von den übrigen SQL-Anweisungen im Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“ beschrieben.
- CALL-DML-Anwendungen
  Das Handbuch richtet sich an den CALL-DML-Programmierer und beschreibt die Sprachelemente der CALL-DML-Schnittstelle sowie das Erstellen von CALL-DML-Programmen.
- Datenbankbetrieb
  Dieses Handbuch richtet sich an den Systemverwalter und beschreibt den Datenbankbetrieb. Der Inhalt umfasst das Starten und Beenden von DBH und SESDCN sowie deren Ladeoptionen und Administrationsanweisungen. Auch die Dienstprogramme, die während des Datenbankbetriebs benötigt werden, sind in diesem Handbuch beschrieben.
- Utility-Monitor
  Das Handbuch beschreibt die Bedienung und die Funktionalität des Utility-Monitors. Der Utility-Monitor ist ein Bestandteil von SESAM/SQL und bietet eine menügesteuerte Oberfläche, um eine Datenbank über SQL-Anweisungen aufzubauen, zu laden, zu sichern und zu rekonstruieren. Auch die Abfrage von Metadaten wird über den Utility-Monitor komfortabel angeboten.
- Meldungen
  Das Meldungshandbuch enthält Informationen über den Aufbau und Abruf der Meldungen des Datenbanksystems SESAM/SQL-Server und der Verteilkomponente SESAM/SQL-DCN. Die SQLSTATEs und CALL-DML-Statusmeldungen sind hier vollständig aufgeführt.

- Performance
  Das Handbuch richtet sich an den erfahrenen Anwender von SESAM/SQL. Es beschreibt, wie der Anwender Performance-Engpässe im Leistungsverhalten von SESAM/SQL erkennen und mit welchen Maßnahmen er das Systemverhalten beeinflussen kann.

Das Erstellen von ESQL-COBOL-Programmen ist im folgenden Handbuch beschrieben:

- ESQL-COBOL Benutzerhandbuch

Der Remote-Zugriff mit SESAM-DBAccess ist im folgenden Handbuch beschrieben:

- SESAM-DBAccess

Zur Suche nach Begriffen ist dem Produkt SESAM/SQL-Server ein PDF-Index der SESAM/SQL-Handbücher beigepackt.

Beispieldatenbank

Zum Lieferumfang von SESAM/SQL-Server gehört die Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB. Diese Bibliothek enthält alle Bestandteile, die Sie benötigen, um die Handbuchbeispiele selber auszuprobieren.

Hierzu zählen:

- Readme-Datei mit einer Übersicht über enthaltene Dateien
- Startprozeduren zum Starten der benötigten SESAM/SQL-Programme
- Dateien der aufgebauten Beispieldatenbank AUFTRAGKUNDEN
- Anweisungsdateien und ESQL-COBOL-Programme mit Ablaufbeispielen für wichtige Datenbank-Anweisungen

Handbuchbeispiele werden durch nebenstehendes Symbol gekennzeichnet, wenn sie als Bestandteil einer Anweisungsdatei oder eines ESQL-COBOL-Programms in der Bibliothek enthalten sind.

Die Beispieldatenbank wird beschrieben im Kapitel „Beispieldatenbank“ auf Seite 55.

1.2 Zielsetzung und Zielgruppen des Handbuchs

Das vorliegende Handbuch beschreibt vorwiegend Grundlagen, Konzepte und Zusammenhänge und wendet sich daher an jeden SESAM/SQL-Anwender.

Zusätzlich richtet sich dieses Handbuch an alle Interessierten, die sich einen Überblick über das Datenbanksystem SESAM/SQL sowie seine Einsatzmöglichkeiten und Zusatzprodukte verschaffen wollen.

Zum Verständnis sind allgemeine Datenbank- und SQL-Kenntnisse sowie Grundkenntnisse des Betriebssystems BS2000 und des Universellen Transaktionsmonitors openUTM von Vorteil.

1.3 Konzept des Handbuchs

Das vorliegende Basishandbuch dient dazu, SESAM/SQL kennenzulernen sowie Grundlagen, Konzepte und Zusammenhänge zu vermitteln.

Alle weiteren SESAM/SQL-Handbücher bauen auf dem Inhalt dieses Handbuchs auf. Deshalb sollten Sie dieses Handbuch möglichst von Anfang an lesen. Das vorliegende Handbuch beschreibt:

- die Leistungsmerkmale des Datenbanksystems SESAM/SQL und informiert über Einsatzmöglichkeiten auch im Zusammenspiel mit Umfeldprodukten
- den SQL-Funktionsumfang von SESAM/SQL im Überblick
- den Umgang mit Utility-Anweisungen
- die Möglichkeiten, SESAM/SQL-Datenbanken vor unberechtigtem Zugriff zu schützen
- das Sicherungskonzept und das Media-Recovery von SESAM/SQL
- Grundsätzliches zum Datenbankbetrieb einschließlich verteilter Verarbeitung mit SESAM/SQL-DCN
- die Arbeitsschritte, die zum Aufbauen, Laden und Warten von SESAM/SQL-Datenbanken erforderlich sind
- die Zusammenarbeit von SESAM/SQL mit openUTM
- die Fachwörter von SESAM/SQL

Auf den Inhalten des Basishandbuchs aufbauend enthalten die weiteren Handbücher zu SESAM/SQL im Wesentlichen alphabetisch sortierte Anweisungsbeschreibungen und Spezialthemen.

Readme-Datei

Funktionelle Änderungen der aktuellen Produktversion und Nachträge den Handbüchern von SESAM/SQL-Server entnehmen Sie bitte ggf. der produktspezifischen Readme-Datei.

Readme-Dateien stehen Ihnen online bei dem jeweiligen Produkt zusätzlich zu den Produkthandbüchern unter *http://manuals.ts.fujitsu.com* zur Verfügung. Alternativ finden Sie Readme-Dateien auch auf der Softbook-DVD.

Informationen unter BS2000

Wenn für eine Produktversion eine Readme-Datei existiert, finden Sie im BS2000-System die folgende Datei:

`SYSRME.<product>.<version>.<lang>`

Diese Datei enthält eine kurze Information zur Readme-Datei in deutscher oder englischer Sprache (<lang>=D/E). Die Information können Sie am Bildschirm mit dem Kommando `SHOW-FILE` oder mit einem Editor ansehen.
Das Kommando `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` zeigt, unter welcher Benutzerkennung die Dateien des Produkts abgelegt sind.

Ergänzende Produkt-Informationen

Aktuelle Informationen, Versions-, Hardware-Abhängigkeiten und Hinweise für Installation und Einsatz einer Produktversion enthält die zugehörige Freigabemitteilung. Solche Freigabemitteilungen finden Sie online unter *http://manuals.ts.fujitsu.com*.

1.4 Änderungen gegenüber dem Vorgänger-Handbuch

In folgender Tabelle sind die wichtigsten Änderungen aufgeführt. Außerdem werden jeweils das Handbuch und das Kapitel genannt, in dem die Änderung beschrieben wird. Wird ein Thema in mehr als einem Handbuch beschrieben, dann wird zuerst das Handbuch aufgeführt, in dem das Thema vollständig beschrieben wird. In der Spalte „Handbuch“ bedeuten die Einträge:

Basis	Basishandbuch	SB T1	Sprachbeschreibung Teil 1
SB T2	Sprachbeschreibung Teil 2	DBB	Datenbankbetrieb
Utimon	Utility Monitor	Perform	Performance

Thema	**Handbuch**	**Kapitel**
SQL-Sprachbeschreibung		
Neue Themen: Routinen, User Defined Functions (UDF). UDF als SQL-Objekt. Neue Abschnitte „Routinen“ und „SQL-Anweisungen zur Gestaltung und Verwaltung von Routinen“. UDF als Funktion Neuer Abschnitt Privilegien von Routinen. Routinen als Teil der Schema-Erstellung.	Basis	Kap.4.1 Kap.4.1.8 Kap.4.2.3 Kap.4.3.5 Kap.6.3.3 Kap.9.1.2.9
Neue Pragmas DEBUG ROUTINE und DEBUG VALUE. Neue Annotationen VOLATILE und IMMUTABLE. Neu gestaltetes Kapitel „Routinen“. Neue Kontrollanweisungen in Routinen. Selbst definierte SQLSTATES und Diagnoseinformationen in Routinen.	SB T1	Kap.3.3 Kap.7 Kap.8.2
Neue SQL-Anweisungen zur Verwaltung von UDFs: CREATE FUNCTION, DROP FUNCTION und RETURN. Neue SQL-Kontrollanweisungen in Routinen: CASE, FOR, ITERATE, REPEAT und WHILE. Neue SQL-Diagnoseanweisungen in Routinen: GET DIAGNOSTICS, SIGNAL und RESIGNAL. Privileg EXECUTE für UDFs bei GRANT und REVOKE. Anpassung der Anweisungen ALTER und DROP TABLE, CREATE und DROP SCHEMA, CREATE und DROP PROCEDURE, DROP TABLE und DROP VIEW an Routinen.		
Neue Views für Routinen im INFORMATION_SCHEMA: ROUTINE_ROUTINE_USAGE, VIEW_ROUTINE_USAGE. Neue Views für Routinen im SYS_INFO_SCHEMA: SYS_ROUTINE_ROUTINE_USAGE, SYS_VIEW_ROUTINE_USAGE.		Kap.9.1 Kap.9.2
Neu gestalteter Abschnitt: Einsatz von Routinen	Perform	Kap.3.4

Tabelle 1: Änderungen von SESAM/SQL-Server V9.0 gegenüber V8.0 (Teil 1 von 3)

Thema	Handbuch	Kapitel
Autonome Transaktionen, neues Pragma AUTONOMOUS TRANSACTION neue Auftraggeber-Identifikation AUTTRAN.	Basis SB T1 DBB	Kap.4.4 Kap.3.3 Kap.5.2.3.3
IN-Prädikat mit nur einem Listenelement	SB T1	Kap.5.3
Einfacher CASE-Ausdruck mit einer Liste von Ausdrücken bei WHEN	SB T1	Kap.5.5.2
Korrelationsnamen: Die Zeichenfolge DO wird in der FOR-Anweisung verwendet. Sie darf nicht mehr als Korrelationsname angegeben werden. Sie können aber den Spezialnamen "DO" verwenden.	SBT1	Kap.3.4 Kap.6.1 Kap.8.2
Utility-Monitor		
Die Zeilenlänge der Anweisungsdatei ist nicht mehr begrenzt. Verwendung von EDT V17.0.	Utimon	Kap.3.3
Erweiterung der Maske INF im Utility-Monitor um Informationen über Routinen auszugeben. Geändete Masken: INF.9, INF.9.3, INF.9.4. Neue Masken: INF.9.3.7, INF.9.3.15, INF.9.4.7, INF.9.9, INF.9.9.1 bis INF.9.9.10. Die Nummerierung der Funktionen in der Maske 9.3 (BASETABLES) wurde inkompatibel geändert. Die neuen Informationen können auch mit CMD INF abgefragt werden.	Utimon	Kap.5.4
Erweiterung der Maske SNF im Utility-Monitor um Informationen über Routinen auszugeben. Neue Masken: SNF.24 bis SNF.29. Die neuen Informationen können auch mit CMD INF abgefragt werden.	Utimon	Kap.5.4
DBH-Administration		
Neue Konnektionsmodul-Parameter: SQLOPT-KEEP-JOIN-ORDER und SQLOPT-PREFERRED-JOIN-METHOD	Basis	Kap.8.2.2
Informationen über Live Migration im SYS_INFO_SCHEMA: neuer View SYS_ENVIRONMENT	SBT1	Kap.9.2
Informationen über „teure“ DML-Anweisungen im SYS_INFO_SCHEMA: neuer View SYS_DML_RESOURCES	SBT1 Perform	Kap.9.2 Kap.2.1 Kap.3.3 Kap.5.1
Remote Zugriff auf SESAM/SQL-Datenbanken		
Remote Datenbankzugriff mit dem PDO-Client über die PHP-Schnittstelle	Basis DBB	Kap.2 Kap.5.2.3.2

Tabelle 1: Änderungen von SESAM/SQL-Server V9.0 gegenüber V8.0 (Teil 2 von 3)

Thema	Handbuch	Kapitel
Performance		
Neues Pragma KEEP JOIN ORDER im SQL-Optimizer	Perform	Kap.2.3
Fachwörter und Masterindex		
Das Handbuch „Fachwörter und Masterindex“ wird nicht fortgeführt. Die Fachwörter wurden in das Basishandbuch integriert. Zur Suche nach Begriffen wird dem Produkt ein PDF-Index der SESAM/SQL-Handbücher beigepackt.	Basis	Kap. Fachwörter
Beispieldatenbank		
Die Beispieldatenbank wurde auf aktuellen Stand gebracht. Neue Anweisungsdateien INSTR.AUFTRAGKUNDEN.090 und INSTR.AUFTRAGKUNDEN.ROUTINES	Basis	Kap.3

Tabelle 1: Änderungen von SESAM/SQL-Server V9.0 gegenüber V8.0 (Teil 3 von 3)

1.5 Darstellungsmittel

Wegen der häufigen Nennung der Bezeichnungen, werden der Einfachheit und Übersichtlichkeit halber folgende Abkürzungen gebraucht:

- **BS2000-Server** für die Server mit /390-Architektur und die Server mit x86-Architektur. Diese Server werden mit dem entsprechenden BS2000-Betriebssystem betrieben.
- **/390-Server** für die Server Unit /390 der FUJITSU Server BS2000 SE Serie und die Business Server der S-Serie
- **x86-Server** für die Server Unit x86 der FUJITSU Server BS2000 SE Serie und die Business Server der SQ-Serie (x86-64-Architektur)

Die Zeichenfolge `<ver>` bezeichnet die aktuelle Version, wenn die Angabe sonst Versionsunabhängig ist.

In diesem Handbuch verwenden wir folgende Darstellungsmittel:

—————— ——————	Syntaxdefinitionen; Fortsetzungszeilen innerhalb von Syntaxdefinitionen sind eingerückt.
GROSS	SQL-Schlüsselwörter
unterstrichen	Voreinstellungen
fett	Hervorhebung im Fließtext
kursiv	Variablen in Syntaxdefinitionen und im Fließtext
`Schreibmaschinenschrift`	Programmtext in Syntaxdefinitionen und in Beispielen
`::=`	Definitionszeichen Die Angabe auf der rechten Seite von ::= definiert die Syntax für das Element auf der linken Seite.
[ ]	Optionale Angaben. Die eckigen Klammern sind Metazeichen, die in einer SQL-Anweisung nicht angegeben werden.
{ } (mehrzeilig)	Alternative Angaben in Syntaxdefinitionen. Jede Zeile enthält eine Alternative. Die geschweiften Klammern sind Metazeichen, die in einer SQL-Anweisung nicht angegeben werden.
{ } (einzeilig)	Zusammenfassung von Klauseln in Syntaxdefinitionen, die gemeinsam wiederholt werden können. Die geschweiften Klammern sind Metazeichen, die in einer SQL-Anweisung nicht angegeben werden.
,...	In Syntaxdefinitionen bedeutet diese Schreibweise, dass die vorausgehende Angabe beliebig oft, durch Komma getrennt, wiederholt werden kann. Wird keine Wiederholung angegeben, fällt auch das Komma weg.
...	In Syntaxdefinitionen bedeuten die Punkte, dass die vorausgehende Angabe beliebig oft wiederholt werden kann. In Beispielen bedeuten die Punkte, dass die restlichen Teile für das Beispiel ohne Bedeutung sind. Die Punkte sind Metazeichen, die in einer SQL-Anweisung nicht angegeben werden.
	Hinweise auf besonders wichtige Informationen
	Warnhinweise

2 Relationales Datenbanksystem SESAM/SQL

SESAM/SQL ist ein relationales Datenbanksystem, das sich in einer Vielzahl von Anwendungen bewährt hat. Es wird von mehreren hunderttausend Anwendern für die unterschiedlichsten Problemstellungen in den Bereichen Wirtschaft, Verwaltung und Wissenschaft genutzt.

Die besondere Stärke von SESAM/SQL ist der Einsatz in Online Transaction Processing-Anwendungen (OLTP-Anwendungen). Es bestehen heute zahlreiche SESAM/SQL-Installationen, bei denen mehrere tausend Benutzer gleichzeitig auf gemeinsame Datenbestände zugreifen. Teilweise werden Tabellen im Gigabyte-Bereich mit bis zu 250 Millionen Datensätzen verwendet. Damit bietet SESAM/SQL ein nach oben offenes Leistungsspektrum. Der Anwender hat die Sicherheit, dass SESAM/SQL auch ein starkes Wachstum der Datenvolumina und Benutzerzahlen bewältigen kann.

SESAM/SQL erfüllt alle Anforderungen, die heute an ein modernes Datenbanksystem gestellt werden, siehe Seite 13.

In den folgenden Abschnitten sind die relationalen Eigenschaften, die Leistungsmerkmale sowie das Produktumfeld von SESAM/SQL beschrieben. Sie erhalten einen umfassenden Überblick über die Zugriffs- und Einsatzmöglichkeiten von SESAM/SQL.

2.1 Vorteile des relationalen Modells

SESAM/SQL dient der Verwaltung und Bearbeitung von Anwenderdaten, deren Struktur dem relationalen Modell folgt.
Das relationale Modell wurde ursprünglich von E.F.Codd 1970 mit Hilfe der Relationenalgebra formuliert und in der Folgezeit mehrfach präzisiert und erweitert (z.B. Codd „The relational Model for Database Management Version 2“, 1990).

Vereinfacht kann man das relationale Modell wie folgt charakterisieren:

- Kennzeichnend für das relationale Modell ist die Darstellung der Datenstrukturen in Form von Tabellen, die sich aus Zeilen (Datensätzen) und Spalten zusammensetzen. Jede relationale Datenbank setzt sich auf logischer Ebene aus Tabellen zusammen. Beziehungen zwischen Tabellen werden vom Anwender über den Inhalt von Spalten hergestellt. Diese Darstellung ist intuitiv verständlich und ermöglicht es auch Endbenutzern, die Struktur einer Datenbank nachzuvollziehen.
- Die Datenmanipulationen sind im relationalen Modell mengenorientiert, d.h. auf Satzmengen und nicht auf einzelne Sätze ausgerichtet. Selbst komplexe Operationen lassen sich aus einfachen, mengenorientierten Grundoperationen zusammensetzen.
- Relationale Datenbanksysteme gewährleisten logische und physikalische Datenunabhängigkeit.
  Logische Datenunabhängigkeit bedeutet, dass ein Benutzer seine Sicht der Daten erzeugen oder ändern kann, ohne dass die Gesamtstruktur der Daten geändert werden muss. Werden neue Spalten oder Tabellen angelegt, dann müssen bestehende Programme nicht geändert werden. Löschen oder Ändern von Spalten erfordert nur eine Anpassung der Programme, die auf diese Spalten zugreifen.
  Physikalische Datenunabhängigkeit bedeutet, dass die logische und physikalische Struktur einer Datenbank voneinander unabhängig sind. Logische Beziehungen zwischen Datenstrukturen können durch Verknüpfen von Spalten formuliert werden und erfordern keine Angaben über die interne Speicherstruktur einer Datenbank. Anwenderprogramme müssen bei einer Änderung der physikalischen Datenstruktur nicht geändert werden.

SESAM/SQL ermöglicht dem Anwender die vollständige Nutzung der genannten Vorteile des relationalen Modells.

2.2 SQL-Schnittstelle von SESAM/SQL

Seit der ersten Formulierung des relationalen Modells durch Codd (1970) wurde an der Entwicklung einer Datenbanksprache gearbeitet, die am relationalen Modell ausgerichtet ist. Seit Anfang der 80er Jahre existierten erste kommerzielle Implementierungen einer solchen Sprachschnittstelle unter dem Namen SQL (Structured Query Language). SQL wurde erstmals 1987 von der International Organization for Standardization normiert (Standard ISO/IEC 9075:1987).
Heutzutage ist SQL die am meisten verbreitete Sprache zur Bearbeitung von relationalen Datenbanken. In der Praxis wird ein Datenbanksystem als relational bezeichnet, wenn es die SQL-Schnittstelle unterstützt.

Analog zum relationalen Modell erfolgt in SQL die Abbildung der Datenstruktur auf Tabellen. Darin werden Daten abgefragt, eingefügt, verändert oder gelöscht. Über die Verknüpfung von Tabellen bzw. Ergebnismengen mehrerer Abfragen lassen sich komplexe Datenbankoperationen formulieren.
SQL ist eine in den Grundzügen leicht erlernbare und gleichzeitig für komplexe Anwendungen geeignete Sprache für relationale Datenbanken. Die Formulierung der Datenbankoperationen in SQL lehnt sich an die englische Sprache an. SQL besitzt Sprachmittel für einfache, mengenorientierte Grundoperationen, auf die sich alle Datenmanipulationen zurückführen lassen.
Im Gegensatz zu den prozeduralen Sprachen nicht-relationaler Datenbanksysteme ist SQL deskriptiv ausgelegt. Das bedeutet, der Anwender beschreibt in einer mengenorientierten Form - wie im relationalen Modell - das Ergebnis einer Datenbankoperation und nicht die zu diesem Ergebnis führenden Schritte. Eine einzelne SQL-Anweisung kann so eine Vielzahl von Operationen auf der Datenbank bewirken, für die bei satzorientierter Verarbeitungsweise viele Anweisungen notwendig wären.

SESAM/SQL basiert auf dem **Standard ISO/IEC 9075:<jahr>**, kurz **SQL-Norm** genannt. Der aktuelle Standard ist ISO/IEC 9075:2008, kurz SQL08 genannt.

SESAM/SQL verfügt über umfangreiche, standardkonforme SQL-Sprachmittel u.a. zur

- Datendefinition einschließlich der Verwaltung von Zugriffsrechten
- Datenmanipulation
- Transaktionsverwaltung
- dynamischen SQL

Der Standard ISO/IEC 9075:1992 unterscheidet die drei Stufen der Normkonformität „Entry Level“, „Intermediate Level“ und „Full Level “. SESAM/SQL deckt den Entry Level vollständig ab sowie den Intermediate Level und den Full Level in wichtigen Funktionen.

Der aktuelle Standard hat diese Einteilung in drei Stufen nicht übernommen. Es wurde Core SQL als Sprachkern eingeführt. Core SQL ist eine echte Obermenge zu Entry SQL und wird von SESAM/SQL vollständig unterstützt.

Neben der durch die Norm festgelegten SQL-Funktionalität kennt SESAM/SQL eigene SQL-Sprachmittel für noch nicht genormte Funktionen, zum Beispiel

- zur Behandlung multipler Spalten
- zur Verwaltung der Speicherstrukturen
- zur Verwaltung von Benutzereinträgen
- zum Ausführen von sogenannten Utility-Anweisungen.
  Utility-Anweisungen sind Anweisungen in SQL-Syntax, die Utility-Funktionen für die Datenbankverwaltung wie Generieren, Laden, Entladen, Kopieren und Reorganisieren von Datenbanken bereitstellen.

Beim Übersetzen von SQL-Programmen kann der Anwender den SQL-Sprachumfang auswählen; damit wird die Entwicklung portierbarer Anwendungen unterstützt.

SQL-Anweisungen können bei SESAM/SQL in mehreren Umgebungen eingesetzt werden:

- innerhalb der Wirtssprache COBOL als eingebettete SQL-Anweisungen (embedded SQL=ESQL), siehe Seite 47.
- innerhalb von DRIVE, einer Programmiersprache der 4. Generation, siehe Seite 46.
- über den Utility-Monitor, der eine menügesteuerte Oberfläche für die Datenbankverwaltung bietet, siehe Seite 37.
- über die von Microsoft definierte ODBC-Schnittstelle (Open Database Connectivity) und das Partnerprodukt ODBC-Rocket der Firma gfs Hamburg in Client-Anwendungen auf UNIX-Systemen und Windows-Systemen, siehe Seite 51.
- über die von Microsoft definierte ADO-Technologie (ActiveX Data Objects) und der ADO.NET-Schnittstelle in Client-Anwendungen auf Windows-Systemen, siehe Seite 51.
- über die JDBC-Schnittstelle für den Zugriff auf SESAM/SQL aus Java-Programmen bzw. Java-Applets auf beliebigen Plattformen, siehe Seite 52.
- über den PDO-Treiber von SESM/SQL für die PHP-Schnittstelle (**P**HP: **H**ypertext **P**rocessor) auf einem Apache Webserver in einer Application Unit unter Linux eines FUJITSU Server BS2000 SE Serie.

SESAM/SQL verwendet SQL-Sprachmittel zur Datendefinition und Datenmanipulation in standardkonformer Weise und stellt alle wesentlichen Funktionen zur Datenbankverwaltung in SQL-Syntax zur Verfügung. Mit SQL liegt somit eine standardisierte, einheitliche Sprache für unterschiedliche Benutzergruppen wie Endbenutzer, Anwendungsprogrammierer und Datenbankverwalter vor, die das Erstellen portierbarer Datenbank-Anwendungen erleichtert. Mit dem Utility-Monitor als Bestandteil von SESAM/SQL (siehe Seite 37) steht eine komfortable Oberfläche für alle wichtigen Aufgaben der Datenbankverwaltung zur Verfügung.

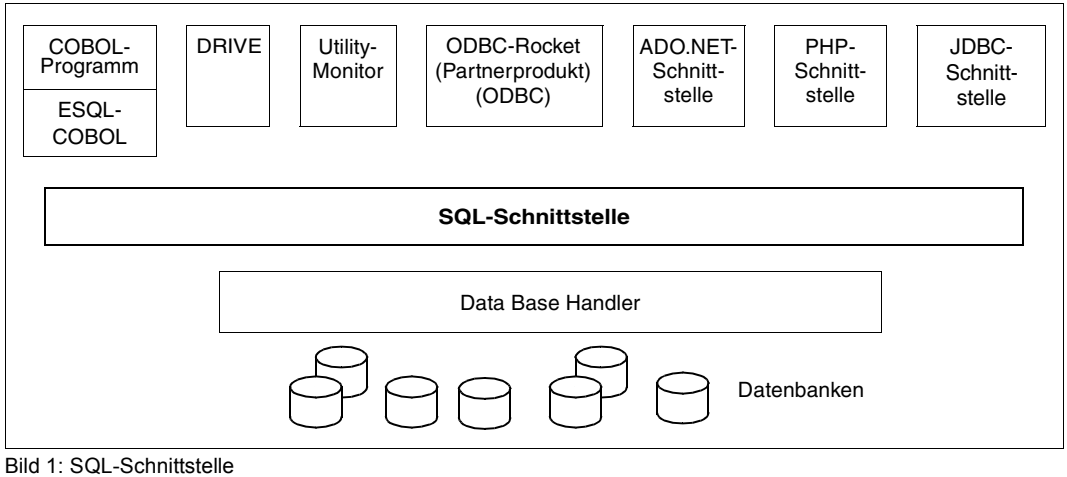


Bild 1: SQL-Schnittstelle

2.3 Unicode-Konzept in SESAM/SQL

Dem Unicode-Zeichensatz kommt im Zuge der zunehmenden Internationalisierung auch in BS2000 und seinen Anwendungen eine entscheidende Bedeutung zu. Nähere Informationen dazu finden Sie im Handbuch „Unicode in BS2000“.

Die Unicode-Unterstützung in BS2000 ist eingebettet in das bereits vorhandene Konzept der codierten Zeichensätze (Coded Character Set, CCS), siehe Handbuch „XHCS (BS2000)“.

Voraussetzung für die Verwendung von Unicode ist eine dementsprechende BS2000-Systemumgebung. Derzeit sind für SESAM/SQL die Produkte ESQL-COBOL, COBOL, CRTE und XHCS relevant, siehe die Freigabemitteilung zu SESAM/SQL-Server.

Das Konzept der Unicode-Unterstützung in SESAM/SQL erlaubt die Verwendung von Unicode-Zeichen in den Spalten von Tabellen und berücksichtigt codierte Zeichensätze in Datenbanken, in Ein-/Ausgabedateien und für Anwenderprogramme. Dieses Konzept wirkt sich auf die SQL-Sprachbeschreibung, die Utility-Funktionen und die SESAM/SQL-Anwenderprogramme aus.

EBCDIC-Zeichensätze

Der Standardzeichensatz im BS2000 ist EBCDIC.DF.03IRV (CCS-Name `EDF03IRV`), ein 7-Bit-Zeichensatz, dessen Zeichenvorrat dem ASCII-7-Bit-Zeichensatz entspricht, erweitert um den zweiten Steuerzeichenblock von ISO8859-1.

EDF03IRV umfasst nur 191 Zeichen, davon 95 abdruckbare Zeichen.
Die 8-Bit-Zeichensätze EDF04x (x=1, 2, 3, 4, 5, 7, 8, 9, F) entsprechen in ihrem Zeichenvorrat den entsprechenden Zeichensätzen ISO8859-x.
Alle EBCDIC-Zeichensätze enthalten denselben EBCDIC-Kern (79 Zeichen). Die 8-Bit-Zeichensätze enthalten außerdem sprachspezifische Zeichen, z.B. griechische Zeichen in EDF047. Für die Anwendung der EBCDIC-Zeichensätze mussten in SESAM/SQL bis V4.0 keine Besonderheiten beachtet werden. Der codierte Zeichensatz der Datenbank (`CODE_TABLE`) hatte Kommentarcharakter.

Zur binären Darstellung (Codierung) eines EBCDIC-Zeichens genügt ein Byte.
Die Länge eines EBCDIC-Zeichens ist demzufolge ein Byte.
Aus Kompatibilitätsgründen zum Unicode-Zeichensatz wird auch für EBCDIC-Zeichenketten die Angabe der Länge in Code Units (1 Code Unit = 1 Zeichen = 1 Byte) definiert.

Vergleich und Sortierung von Werten bzw. Spalten mit alphanumerischen (EBCDIC-) Datentypen erfolgen binär:
lateinische Kleinbuchstaben (a-z) < lateinische Großbuchstaben (A-Z) < Ziffern (0-9)

Das Wort „alphanumerisch“ drückt in der Handbuchreihe SESAM/SQL die Zugehörigkeit zu einem EBCDIC-Zeichensatz aus, z.B. alphanumerischer Datentyp, alphanumerischer Wert, alphanumerisches Literal.

Unicode-Zeichensatz

Unicode fasst alle weltweit bekannten Textzeichen in einem einzigen Zeichensatz zusammen. Zudem ist Unicode unabhängig von unterschiedlichen Herstellern, Systemen und Ländern.

In Unicode wird jedem Zeichen eine Nummer, der sogenannte **Code Point**, zugeordnet. Ein Unicode Code Point wird im Allgemeinen in der Form `U+n` angegeben, wobei `n` aus 4 bis 6 sedezimalen Ziffern besteht. Beispiel: das Euro-Zeichen €: `U+0020AC`.
Der Unicode-Zeichensatz umfasst über eine Million Code Points.

Für die Darstellung von Unicode-Werten oberhalb `FFFF` (sedezimal) werden sogenannte Surrogate Pairs verwendet:

Code Point-Bereich	UTF-16-Codierung (2 Byte)	Bemerkungen
	`D800 ... DBFF`	Leading Surrogate
	`DC00 ... DFFF`	Trailing Surrogate
`U+010000 ... U+10FFFF`	Surrogate Pair	Leading Surrogate gefolgt von Trailing Surrogate

Tabelle 2: Surrogate-Darstellung

„Noncharacters“ sind Code Points, die in Unicode für interne Zwecke reserviert sind. Sie dürfen in SESAM/SQL nicht in Unicode-Zeichenketten verwendet werden.

Die folgende Tabelle zeigt die Code Point-Bereiche der Noncharacters und wie diese Zeichen in UTF-16 codiert sind.

Code Point-Bereich	UTF-16-Codierung (2 Byte)	Bemerkungen
`U+00FDDx, U+00FDEx`	nicht zulässig	32 Code Points
`U+0xFFFE, U+0xFFFF`	nicht zulässig	32 Code Points
`U+01FFFE, U+10FFFF`	nicht zulässig	2 Code Points

Tabelle 3: Unicode-Noncharacters (x ist eine Sedezimalziffer)

Code Points werden für den Einsatz in der Datenverarbeitung auf verschiedene Arten in Bytes dargestellt (codiert). Das Unicode-Konsortium definiert dafür drei verschiedene Codierungsformen („encoding forms“): UTF-8, **UTF-16** und UTF-32.
Die Länge von Unicode-Zeichenketten in der jeweiligen Codierungsform wird dabei in **Code Units** angegeben. Beschränkt man sich bei der Codierungsform UTF-16 auf das sogenannte „Basic Multilingual Plane“ (BMP, entspricht UCS-2) so ist 1 Code Unit = 2 Byte.

SESAM/SQL verwendet in den Datenbanken zur Darstellung von Unicode-Zeichen die Codierungsform UTF-16.

Zur binären Darstellung (Codierung) eines UTF-16-Zeichens in SESAM/SQL werden also genau zwei Byte benötigt, d.h. eine Code Unit in UTF-16. Vergleich und Sortierung von Werten bzw. Spalten mit Unicode-Datentypen erfolgen (bezogen auf UTF-16) binär: Ziffern (0-9) < lateinische Großbuchstaben (A-Z) < lateinische Kleinbuchstaben (a-z).

Das Wort „National“ drückt in der Handbuchreihe SESAM/SQL die Zugehörigkeit zum Unicode-Zeichensatz aus, z.B. National-Datentyp, National-Wert, National-Literal.

Der technische Report des Unicode-Standards beschreibt in Anlehnung an die Codierungsform UTF-8 für ASCII-Server eine UTF-EBCDIC-Codierung für EBCDIC-Server.
Die Zeichen von UTF-8, die in einem Byte dargestellt werden, entsprechen den Zeichen des ASCII-7-Bit-Zeichensatzes. Analog entsprechen die Zeichen von UTF-EBCDIC, die in einem Byte dargestellt werden, dem Standardzeichensatz EDF03IRV von BS2000.
Der Vorteil von UTF-EBCDIC (BS2000, CCS-Name `UTFE`) liegt darin, daß alle Systemprogramme von BS2000, die sich traditionell auf den Zeichenvorrat EDF03IRV beschränken (z.B. der BS2000-Kommandoprozessor), ohne Anpassungen auch Zeichenketten in der Codierungsform UTF-EBCDIC bearbeiten können.

Die Codierung von UTF-EBCDIC in BS2000 ist proprietär und entspricht nicht der Codierung anderer Hersteller.

Unterstützung von Unicode in SESAM/SQL

Zur Unterstützung von Unicode gibt es in SESAM/SQL die Datentypen NATIONAL CHARACTER (NCHAR) und NATIONAL CHARACTER VARYING (NVARCHAR).
In Spalten mit diesen Datentypen können Unicode-Daten gespeichert werden.

Die Daten werden in SESAM/SQL in der Codierungsform UTF-16 (im Umfang der BMP) gespeichert. Da in der internen Darstellung von SESAM/SQL für ein UTF-16-Zeichen zwei Byte Speicherplatz (eine Code Unit) benötigt werden, ergeben sich für die Unicode-Datentypen folgende maximale Längen:

- NCHAR: 128 Zeichen (256 Byte)
- NVARCHAR: 16000 Zeichen (32000 Byte)

Die Unicode-Datentypen können in Operationen wie Vergleich, Konkatenation und Zuweisung verwendet werden.

Alle bisher in SESAM/SQL vorhandenen Funktionen für CHARACTER (VARYING) gibt es auch für die neuen Datentypen NATIONAL CHARACTER (VARYING). Hierbei müssen die jeweiligen Randbedingungen berücksichtigt werden.

Die neuen Datentypen werden auch in den betreffenden DDL- und Utility-Anweisungen unterstützt, siehe die Anweisungen CREATE TABLE und ALTER TABLE in der „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“ bzw. CREATE CATALOG, ALTER CATALOG, LOAD, UNLOAD, EXPORT, IMPORT in der „SQL-Sprachbeschreibung Teil 2: Utilities“.

Codierter Zeichensatz der Datenbank

Zur korrekten Interpretation von (alphanumerischen) Zeichenketten in Spalten mit den Datentypen CHARACTER und CHARACTER VARYING muss SESAM/SQL den codierten (EBCDIC-)Zeichensatz kennen, in dem die Daten codiert sind.

Beim Erzeugen einer neuen Datenbank (mit CREATE CATALOG) bzw. beim Ändern einer bestehenden Datenbank (mit ALTER CATALOG) können Sie deshalb mit dem Parameter `CODE_TABLE` den CCS-Namen für die Datenbank bestimmen bzw. ändern. Der CCS-Name der Datenbank wird in SESAM/SQL bei Konvertierungen zwischen Daten des Typs (VAR)CHAR und N(VAR)CHAR und umgekehrt benutzt. Dies ist z.B. notwendig bei einer entsprechenden Änderung des Datentyps einer Spalte und in einigen Varianten der Utility-Anweisungen LOAD und UNLOAD.

Codierter Zeichensatz der Anwendung

Zur korrekten Interpretation von Zeichenketten muss SESAM/SQL den codierten (EBCDIC-)Zeichensatz kennen, mit dem die Anwendung die Zeichenketten interpretiert.

Mit dem neuen Konnektionsmodul-Parameter `CCSN` (CCS-Name) geben Sie dem independent-DBH bekannt, mit welchem Zeichensatz die Anwendung Zeichenketten interpretiert. Dem linked-in DBH geben Sie dies über die DBH-Option LINKED-IN-ATTRIBUTES bekannt.

SQL-Anweisungen können im DBH nur verarbeitet werden, wenn der CCS-Name der Anwendung mit dem CCS-Namen der Datenbank übereinstimmt oder wenn für die Datenbank kein codierter Zeichensatz verwendet wird (`CODE_TABLE` hat den Wert `_NONE_`).

Aktivierung von Unicode in einer bestehenden Datenbank

In SESAM/SQL-Datenbanken bis einschließlich V4.0 hatte der Parameter `CODE_TABLE` bei CREATE CATALOG Kommentarcharakter und wurde wie angegeben in den Metadaten gespeichert. Da der Parameter nun eine Bedeutung hat, erhält er beim ersten Zugriff auf eine bestehende Datenbank durch SESAM/SQL seit V5.0 den Wert `_NONE_`, d.h. die Datenbank verwendet keinen codierten Zeichensatz.

Zur Nutzung von Unicode muss der Datenbank-Administrator mit der Utility-Anweisung ALTER CATALOG den aktuell verwendeten codierten (EBCDIC-)Zeichensatz für die Datenbank eintragen, z.B. EDF03IRV oder EDF041 für eine deutsche BS2000-Umgebung. Auch für die Anwendungen sollte der codierte Zeichensatz festgelegt werden, siehe oben.

Wenn der Wert `_NONE_` für die Datenbank beibehalten wird, dann sind keine impliziten Konvertierungen von (VAR)CHAR nach N(VAR)CHAR und umgekehrt möglich. Alle Aufträge an die Datenbank, die eine solche Konvertierung benötigen, werden abgewiesen.

Beispiel 1

Eine Tabelle PERSONAL hat die Spalte NACHNAME mit dem Datentyp CHARACTER(40). Der CCS-Name der Datenbank ist EDF041, die Daten sind also mit dem CCS-Namen EDF041 gespeichert. Zur Nutzung von Unicode gibt es zwei Möglichkeiten:

1. Mit der folgenden DDL-Anweisung wird der Datentyp der Spalte NACHNAME in NATIONAL CHARACTER geändert und die in dieser Spalte gespeicherten Daten werden implizit von EDF041 nach UTF16 konvertiert:
   ```
   ALTER TABLE PERSONAL ALTER COLUMN NACHNAME SET NCHAR(40)
   ```
2. Mit der folgenden Anweisung wird eine zusätzliche Spalte definiert:

   ```
   ALTER TABLE PERSONAL ADD COLUMN ALIAS_NACHNAME NCHAR(40)
   ```

   Anschließend werden die Daten explizit von der ursprünglichen Spalte vom Datentyp CHARACTER in die Zielspalte vom Datentyp UTF16 konvertiert. Dabei wird der codierte Zeichensatz EDF041 der Datenbank verwendet:

   ```
   UPDATE PERSONAL
          SET ALIAS_NACHNAME=TRANSLATE(NACHNAME USING CATALOG_DEFAULT)
   ```

Beispiel 2

Eine Tabelle PERSONAL hat die Spalte NACHNAME mit dem Datentyp CHARACTER(40). Der CCS-Name der Datenbank hat den Wert _NONE_. Mit folgender Anweisung wird eine zusätzliche Spalte definiert:

```
ALTER TABLE PERSONAL ADD COLUMN ALIAS_NACHNAME NCHAR(40)
```

Anschließend werden die Daten explizit von der ursprünglichen Spalte vom Datentyp CHARACTER in die Zielspalte vom Datentyp UTF16 konvertiert. Dabei wird der codierte Zeichensatz EDF041 explizit angegeben:

```
UPDATE PERSONAL SET ALIAS_NACHNAME = TRANSLATE(NACHNAME USING EDF041)
```

Folgende Anweisung wird abgewiesen, da der CCS-Name der Datenbank _NONE_ ist und somit die Konvertierung nicht ausgeführt werden kann.

```
UPDATE PERSONAL SET ALIAS_NACHNAME=TRANSLATE(NACHNAME USING CATALOG_DEFAULT)
```

Transliteration, Transcoding

Zum Umwandeln von alphanumerischen Zeichenketten (Datentyp (VAR)CHAR) in Unicode-Zeichenketten (Datentyp N(VAR)CHAR, Zeichensatz UTF-16) und umgekehrt (Transliteration) gibt es in SESAM/SQL die SQL-Funktion TRANSLATE().

TRANSLATE() wandelt auch alphanumerische Zeichenfolgen, die im Unicode-Zeichensatz UTF-EBCDIC (BS2000, CCS-Name UTFE) vorliegen, in den Unicode-Zeichensatz UTF-16 um und umgekehrt (Transcoding).
Siehe die „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“.

Normalisierung

Die Codierung eines Zeichens in Unicode ist nicht eindeutig, d.h. es kann für ein Zeichen mehr als eine Codierung geben.

Ein typisches Beispiel sind die deutschen Umlaute. Beispielsweise hat das Zeichen Ä sowohl den Code Point U+00C4 (composed form) als auch die Code Point-Kombination U+0041 und U+0308 (decomposed form). In normalisierten Darstellungsformen treten diese Unterschiede nicht auf. Wenn zwei normalisierte Zeichenketten unterschiedlich sind, dann sind es auch ihre unterschiedlichen Code Point-Darstellungen.

Nicht-normalisierte Zeichenketten können zu Folgefehlern nach einer Sortierung führen.

In SESAM/SQL bringt die SQL-Funktion NORMALIZE() eine Zeichenkette mit National-Zeichen (Datentyp N(VAR)CHAR) in eine normalisierte Form. Dabei werden nur Zeichen berücksichtigt, die Code Points im Bereich U+0000 bis U+2FFF besitzen. Andere Zeichen, z.B. Surrogates, bleiben unverändert. Siehe die „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“.

Der Vorgang der Normalisierung benötigt CPU-Leistung. Deshalb sollten die Daten einer SESAM/SQL-Datenbank in normalisierter und komprimierter Form vorliegen.

Wenn nicht sichergestellt ist, ob (Eingabe-)Daten in normalisierter Form vorliegen, sollte eine Normalisierung mit der SQL-Funktion NORMALIZE() in die Normalization Form C durchgeführt werden.

Beispiel

Der Code Point U+1ED6 entspricht dem lateinischen Großbuchstaben „O“ mit Zirkumflex und Tilde. Dieses Zeichen kann mit Hilfe dreier Code Points U+00D4 für „Ô“ und U+0303 für Tilde bzw. U+004F für „O“ und U+0302 für Zirkumflex und U+0303 für Tilde erzeugt werden. Auch die Code Point-Sequenz U+00D5 für das „Õ“ mit Tilde und U+0302 für Zirkumflex ergibt dieses Zeichen. Es gibt nur die Regel, dass das Grundzeichen vor den damit verknüpften diakritischen Zeichen steht.

D.h. das Ergebnis der Normalisierungsform C (compose) von NORMALIZE() des lateinischen Großbuchstaben „O“ mit Zirkumflex und Tilde ist U+1ED6, das Ergebnis der Normalisierungsform D (decompose) ist die Code Point-Kombination U+004F, U+0302, U+0303.

In BS2000 werden Normalisierungsfunktionen auch durch die Komponente XHCS angeboten, siehe Handbuch „XHCS (BS2000)“.

Sortierreihenfolge

In den meisten Programmen werden Zeichenketten binär verglichen. Die binäre Sortierreihenfolge der Zeichen ist abhängig von ihrer Codierung:

- Für alle ISO8859- und Unicode-Codierungen gilt:
  Ziffern (1-9) < lateinische Großbuchstaben (A-Z) < lateinische Kleinbuchstaben (a-z)
- Für EBCDIC und UTFE gilt:
  lateinische Kleinbuchstaben (a-z) < lateinische Großbuchstaben (A-Z) < Ziffern (1-9).

Umlaute und diakritische Zeichen sind nicht eindeutig in diese Sortierreihenfolgen eingeordnet. Damit können sich unschöne Sortierreihenfolgen ergeben, z.B. kann in der Sortierung einer Namensliste der Name „Zuse“ vor dem Namen „Öhler“ erscheinen.

Die Unicode-Norm beschreibt einen linguistischen Sortieralgorithmus. Jedem Unicode-Zeichen wird ein Sortierelement (Collation-Element) zugeteilt. Die Reihenfolge nach der die Unicode-Zeichen sortiert werden, wird mit Hilfe dieser Collation-Elemente festgelegt. Die Collation-Elemente werden mittels einer von XHCS gelieferten Tabelle (Default Unicode Collation Table, DUCET) festgelegt. Diese Tabelle enthält eine Wertigkeit des Zeichens auf verschiedenen Ebenen (Leveln). SESAM/SQL unterscheidet drei Ebenen, die in der nachfolgenden Tabelle dargestellt sind. Der Vergleich der einzelnen Zeichen findet jeweils von links nach rechts statt. Der erste Unterschied bestimmt das Vergleichsergebnis.

Vergleichsebene	**Beschreibung**	**Beispiel**
Ebene 1	Grundzeichen (base character)	a < b role < roles < rule
Ebene 2	Akzente (diakritische Zeichen)	A < Å role < rôle < roles
Ebene 3	Groß-/Kleinschreibung	a < A role < Role < rôle

Ebene 1 : Jedem Grundzeichen (a,b,c...) ist eine feste Wertigkeit in der Default Unicode Collation Table zugeordnet. Nachfolgende Zeichen oder diakritische Zusätze haben keinen Einfluß auf die Sortierreihenfolge der Zeichen.

Ebene 2 : Zum Grundzeichen gehört ein diakritisches Zeichen. Ein diakritisches Zeichen ist ein Zusatzzeichen (z.B. Akzent, Strich, Punkt, Häkchen, Tilde), das in, über oder unter einen Buchstaben gesetzt wird, um dessen Aussprache oder Betonung näher zu bezeichnen. Grundzeichen mit diakritischem Zeichen haben auf Ebene 1 die gleiche Wertigkeit wie das zugehörige Grundzeichen ohne diakritisches Zeichen. Auf Ebene 2 haben Zeichen mit diakritischem Zeichen eine höhere Wertigkeit als dasselbe Zeichen ohne diakritisches Zeichen. Ist der gesamte Sortierbegriff ansonsten gleich, wird mit diesem diakritischen Zeichen die Sortierreihnefolge (siehe Beispiel Ebene 2: role < rôle) festgelegt.

Ebene 3: Die Sortierreihenfolge wird durch die Unterscheidung zwischen Groß- und Kleinbuchstaben festgelegt. Großbuchstaben haben eine höhere Wertigkeit als Kleinbuchstaben. Ebene 3 wird nur berücksichtigt, wenn Ebene 1 und Ebene 2 für den gesamten Sortierbegriff gleich sind
(siehe Beispiel Ebene 3: role < Role).

Die Werte für die Collation-Elemente, die unter der Web-Seite des Unicode-Konsortiums veröffentlicht sind, können sich ändern. Diese Tabelle finden Sie z.B. unter der Adresse: *http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt*.

In BS2000 können Sie sich das Collation-Element über XHCS besorgen, siehe auch Handbuch „XHCS (BS2000)“.

Die SQL-Funktion COLLATE() liefert für National-Zeichenketten das entsprechende Collation-Element der Default Unicode Collation Table.
Siehe die „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“.

2.4 CALL-DML-Schnittstelle

Dem Anwender steht neben der SQL-Schnittstelle die leistungsstarke und vielfach bewährte CALL-DML-Schnittstelle zur Verfügung, um Daten abzufragen und zu ändern. SESAM/SQL-Datenbanken können über diese Schnittstelle mit Programmen der Programmiersprachen C, C++, COBOL, PASCAL, Assembler, PL/1, RPG, ALGOL und FORTRAN bearbeitet werden.

Neben den Anweisungen zum Ändern, Löschen und Einfügen von Datensätzen bietet die CALL-DML mengenorientierte Verarbeitung über ein Cursorkonzept, Funktionen zur Transaktionssteuerung und vielfältige Möglichkeiten der Selektion von Daten.

CALL-DML- und SQL-Anweisungen können auch innerhalb eines Anwenderprogramms und einer CALL-DML-Transaktion gemeinsam eingesetzt werden.

Alle Funktionen der CALL-DML sind im Handbuch „CALL-DML Anwendungen“ ausführlich beschrieben, ebenso wie die Werkzeuge, die dem CALL-DML-Programmierer z.B. zum Test zur Verfügung stehen.

2.5 Komfortable Administration

Große Datenbankanwendungen erfordern eine möglichst komfortable Administration des Datenbanksystems, damit die Systemverwalter und Datenbankverwalter auch hohe und wachsende Abwicklungsvolumina problemlos bewältigen. SESAM/SQL bietet zahlreiche Funktionen zur Unterstützung der Administration.

Administrationsprogramm SESADM

SESADM ermöglicht die Steuerung des Datenbankbetriebs über eine maskengestützte Oberfläche nach SDF-Konventionen. Mit SESADM kann der Systemverwalter den SESAM/SQL-DBH und die SESAM/SQL-Verteilkomponente SESDCN über eine einheitliche Anwenderschnittstelle administrieren. Die Bedienung von SESADM ist im Handbuch „Datenbankbetrieb“ beschrieben.

Utility-Monitor SESUTI

Mit dem Utility-Monitor als Bestandteil von SESAM/SQL steht für alle Aufgaben der Datenbankverwaltung eine komfortable Oberfläche zur Verfügung, zum Beispiel für das Aufbauen, Laden, Sichern und Rekonstruieren einer SESAM/SQL-Datenbank.

Der Utility-Monitor kann im Dialog- oder im Batchbetrieb (Stapelbetrieb) eingesetzt werden. Die Aktionen lassen sich in eine sogenannte Anweisungsdatei protokollieren, die der Anwender bei Bedarf anpassen und für häufig wiederkehrende Tätigkeiten einsetzen kann.

Im Dialog kann der Datenbankverwalter durch die menügeführte Oberfläche des Utility-Monitors die für alle Aufgaben angebotenen SQL-Sprachelemente verwenden, ohne SQL-Anweisungen zu programmieren.

Im Einzelnen führt der Utility-Monitor die folgenden Gruppen von SQL-Anweisungen aus:

- Anweisungen zur Datendefinition einschließlich der Verwaltung von Zugriffsrechten
- Anweisungen zur Verwaltung von Speicherstruktur und Benutzereinträgen
- Utility-Anweisungen
- Anweisungen zur Datenmanipulation

Außerdem kann sich der Anwender mit dem Utility-Monitor bequem über die Metadaten einer Datenbank wie die Beschreibungen von Basistabellen, Views, Integritätsbedingungen, Zugriffsrechten usw. informieren sowie Angaben zu SESAM-Sicherungsbeständen lesen und ändern.
Es ist ebenfalls möglich, aus dem Utility-Monitor heraus SESADM aufzurufen, um Anweisungen zur Steuerung des Datenbankbetriebs abzusetzen.

Das Handbuch „Utility-Monitor“ beschreibt die Bedienung des Utility-Monitors ausführlich.

Performance-Monitor SESMON

Der Performance-Monitor SESMON informiert den Systemverwalter oder (über SNMP) eine Management-Plattform sehr ausführlich und übersichtlich über die Auslastung von Betriebsmitteln (z.B. Pufferauslastungen, Plattenzugriffe). Dadurch lässt sich das Datenbanksystem optimal auf den jeweiligen Einsatzfall einstellen. SESMON ist auf Seite 336 im Überblick und im Handbuch „Datenbankbetrieb“ ausführlich beschrieben.

Zugang aus dem World Wide Web

Zugang zum Administrationsprogramm SESADM, zum Performance-Monitor SESMON und zum Utility-Monitor SESUTI erhalten Sie auch über einen einheitlichen Zugang aus dem World Wide Web (kurz: WWW oder Web) mit Hilfe des Softwareprodukts WebTransactions (WebTA).

Für den Zugang zu den SESAM-Programmen über das Web benötigen Sie neben dem Softwareprodukt WebTransactions lediglich einen Standard-Browser.

Der Web-Zugang ist in einem eigenen Dokument „WebTA-Zugang für SESAM/SQL“ beschrieben, das mit SESAM/SQL-Server ausgeliefert wird. Sie finden dieses Dokument auch auf unserem Handbuchserver beim Softwareprodukt SESAM/SQL.

Nutzung von HSMS

HSMS (Hierarchisches Speicher Management System) kann zur Bandsicherung in SESAM/SQL verwendet werden. HSMS dient zur Datensicherung und zur Unterstützung der Datenverwaltung auf externen Speichern im BS2000. Insbesondere unterstützt HSMS das System der Miteigentümerschaft des Softwareprodukts SECOS, mit dessen Hilfe Sie Zugriffsrechte auf fremde Benutzerkennungen verwalten können.

Nutzung von ARCHIVE

Das High-Performance-Sicherungssystem ARCHIVE kann zur Bandsicherung in SESAM/SQL verwendet werden. Der Anwender steuert die ARCHIVE-Funktionen über SESAM/SQL-eigene Sprachmittel.
ARCHIVE ist ein Standard-Sicherungssystem im BS2000, das auch für viele andere Anwendungen einheitlich benutzt wird.

E-Mail-Versand wichtiger Informationen der DBH-Session

Der SESAM/SQL-Systemverwalter kann wichtige Informationen der DBH-Session auch automatisch per E-Mail über den Mail-Sender des Softwareprodukts interNet Services versenden lassen.
Das Verfahren ist auf Seite 335 im Überblick und im Handbuch „Datenbankbetrieb“ ausführlich beschrieben.

Protokollierung sicherheitsrelevanter Ereignisse mit SAT

SESAM/SQL protokolliert sicherheitsrelevante Ereignisse mit Hilfe der Komponente SAT (Security Audit Trail) des Softwareprodukts SECOS (Security Control System).
Der SESAM-Systemverwalter kann die SAT-Protokollierung ein- und ausschalten.

2.6 Hohe Leistungsfähigkeit

Als leistungsfähiger Datenbankserver für alle Einsatzbereiche ist SESAM/SQL in der Lage, unterschiedliche Auftragstypen ohne gegenseitige Behinderung auszuführen. Prinzipiell lassen sich zwei Auftragstypen unterscheiden:

- OLTP-Anwendungen
  In OLTP-Anwendungen greift eine hohe Anzahl von Benutzern auf die gleichen Datenbanken und Anwendungen zu. Beispiele sind Auftrags-Abwicklungssysteme, Buchungssysteme und Lagerhaltungssysteme. Man spricht hier oft von „Produktivbetrieb“.
  Die OLTP-Transaktionen bestehen in der Regel aus wenigen Lese- und Schreibanweisungen. Dabei wird eine kleine Anzahl unterschiedlicher Transaktionen sehr oft wiederholt.
  Das Thema OLTP ist auf Seite 45 genauer beschrieben.
- Individuelle Datenverarbeitung (IDV)
  In der individuellen Datenverarbeitung werden häufig sehr komplexe Anfragen an das Datenbanksystem gestellt, bei denen oft ganze Datenbanken gelesen und umfangreiche Berechnungen durchgeführt werden. Als typisches Beispiel sind statistische Auswertungen zu nennen.

Mit modernsten Technologien realisiert SESAM/SQL zusammen mit dem Transaktionsmonitor openUTM einen performanten OLTP-Betrieb mit kurzen Antwortzeiten. Gleichzeitig sorgen Parallelisierungsfunktionen und ein leistungsfähiger Optimizer dafür, dass die IDV den OLTP-Betrieb nicht behindert. Dies ist eine wichtige Voraussetzung für Client/Server-Architekturen, in denen z.B. PC-Anwendungen (Tabellenkalkulation, Textverarbeitung usw.) umfangreiche Anfragen starten, um Daten aufzubereiten und weiterzuverarbeiten (siehe auch Seite 49). Die folgenden Abschnitte beschreiben die wichtigsten Technologien und Funktionen.

Multithreading

Mit der Multithreading-Architektur kann der SESAM/SQL-Data Base Handler (DBH) Aufträge parallel verarbeiten und so die Zeit nutzen, in der Aufträge z.B. auf den Abschluss von Ein- und Ausgabevorgängen warten. Für die Dauer des Ein- und Ausgabevorgangs aktiviert SESAM/SQL dann die Bearbeitung eines anderen durchführbaren Auftrags. Der Durchsatz steigt dadurch erheblich. Ebenso werden langlaufende und komplexe Datenbankabfragen portionsweise abgearbeitet, ohne den OLTP-Betrieb zu behindern. Der DBH nutzt die Anzahl der Threads lastabhängig; nicht alle Threads sind also ständig belegt.

Multitasking

SESAM/SQL-Server gibt es als Standard Edition mit Singletask-Betrieb und als Enterprise Edition, die den Multitask-Betrieb unterstützt.

Mit der Multitasking-Architektur kann der SESAM/SQL-Data Base Handler (DBH) bei höheren Performance-Anforderungen auch mit mehreren Tasks geladen werden. Dadurch lässt sich bei Multiprozessor-Anlagen die DBH-Last auf mehrere Prozessoren verteilen.

Auslagerung CPU-intensiver Aktionen

SESAM/SQL lagert CPU-intensive Aktionen auf sogenannte Service-Tasks aus, wo sie parallel zum eigentlichen DBH-Betrieb ablaufen. Service-Tasks stehen zum Beispiel für CPU-intensive Funktionen der Datenbankverwaltung und für das Sortieren von Zwischenergebnissen zur Verfügung.

Cost Based Optimizer

Wenn ein Anwenderprogramm eine SQL-Anweisung absetzt, erstellt SESAM/SQL einen Zugriffsplan. Dieser beschreibt die Art und die Reihenfolge der einzelnen Auswertungsschritte der SQL-Anweisung. Der Cost Based Optimizer sorgt dafür, dass ein besonders effizienter Zugriffsplan erstellt wird, bei dem möglichst wenig Systemressourcen beansprucht werden (CPU-Zeit, I/O-Zugriffe).

Shared SQL

Der optimierte Zugriffsplan wird im Arbeitsspeicher gehalten und kann von mehreren Benutzern verwendet werden. Besonders bei OLTP-Anwendungen, in denen bestimmte Verarbeitungsschritte häufig wiederholt werden, erhöht sich durch Shared SQL die Leistung entscheidend.

Shared Record Lock

Wenn ein Datensatz gelesen wird, so wird dieser normalerweise für andere Transaktionen gesperrt. Durch den „Shared Record Lock“ ist es möglich, dass andere Transaktionen diesen Satz lesen können. Dadurch sinkt die Zahl der Blockierungen, und es können mehr Transaktionen parallel durchgeführt werden. Die Transaktionsleistung erhöht sich. Das erweiterte Sperrkonzept beeinträchtigt die Transaktionssicherheit nicht, da Shared Record Locks immer nur dann möglich sind, wenn keine Daten verändert werden.

Datenkomprimierung

SESAM/SQL komprimiert die Daten bei der Speicherung automatisch. Sie beanspruchen somit weniger Speicherplatz.

- Es werden nur signifikante Werte abgespeichert, nicht signifikante Werte belegen keinen Speicherplatz.
- Die Datentypen NUMERIC und DECIMAL werden nur in der signifikanten Länge ohne die führenden Nullen abgespeichert.
- Der Datentyp CHARACTER wird nur in der signifikanten Länge abgespeichert, wobei die nachfolgenden Leerzeichen entfallen.

Durch die Komprimierung der Daten auf signifikante Werte kann die Datenbank auf maximale Erfordernisse ausgelegt werden. Damit ist Folgendes problemlos möglich:

- Spalten definieren, für die es nur in wenigen Datensätzen Werte gibt
- bereits beim Einrichten der Datenbank Spalten definieren, die erst für zukünftige Anwendungen infrage kommen
- Spaltendefinitionen beibehalten, auch wenn es keine Werte mehr dafür gibt.

Da die Datensätze wegen der Komprimierung kürzer sind und sich deshalb über weniger Speicherblöcke erstrecken, beschleunigt sich der Zugriff auf die Daten. Zudem können mehr Datensätze im Arbeitsspeicher gehalten werden, wodurch die Anzahl der Plattenzugriffe reduziert wird.

SESAM/SQL-LINK - die Linked-in-Variante von SESAM/SQL

Die Linked-in-Variante ist für die besonders effiziente Verarbeitung eines einzelnen Anwenderprogramms gedacht. Der linked-in DBH wird dabei fest in das Anwenderprogramm eingebunden. Die angeschlossenen Datenbanken sind dem Anwenderprogramm exklusiv zugeordnet. Gegenüber dem Betrieb mit dem independent DBH entfällt bei der linked-in Variante die Zeit für die Kommunikation zwischen Anwenderprogramm und DBH sowie der Aufwand für die Verwaltung von Sperren.

Der Einsatzbereich von SESAM/SQL-LINK liegt in der Abarbeitung von Batchprogrammen mit exklusivem Datenbankzugriff.

Funktionale Abweichungen oder Hinweise, die beim Einsatz des linked-in DBH zu beachten sind, sind innerhalb der SESAM/SQL-Handbücher an der Stelle beschrieben, wo sie für den Anwender wichtig sind.

SESAM/SQL-LINK ist für SX-Server nicht verfügbar.

Global Storage

Die Performance eines Datenbanksystems wird nicht nur von der Prozessorleistung beeinflusst. Gerade bei Datenbanksystemen erfolgen viele Lese- und Schreibzugriffe auf Festplatten, die im Vergleich zu Zugriffen auf den Hauptspeicher relativ langsam sind. Für die BS2000-Systeme wurden deshalb besonders schnelle Speichermedien entwickelt. So bietet zum Beispiel der Global Storage - ein batteriegepufferter Halbleiterspeicher - eine 2000-mal schnellere Zugriffszeit als Magnetplatten. SESAM/SQL kann den Global Storage als Datencache benutzen, wodurch Schreib- und Lesezugriffe auf Platten drastisch beschleunigt werden. Die Gesamtleistung verbessert sich deutlich.

Schubmodus (Block Mode)

Ein Cursor kann mit sogenanntem Schubmodus definiert werden. Im Schubmodus veranlasst die erste Ausführung der FETCH NEXT-Anweisung, dass mehrere Sätze in einen Zwischenpuffer gelesen werden. Der Anwender erhält mit der ersten Ausführung der FETCH-Anweisung nur den ersten übertragenen Satz. Die nachfolgenden Ausführungen der FETCH-Anweisung übertragen jeweils einen Satz aus dem Zwischenpuffer (ohne erneute Task-Task-Kommunikation), bis der Zwischenpuffer leer ist. Eine weitere Ausführung der FETCH-Anweisung führt dann wieder zur Übertragung mehrerer Sätze. Durch diesen Schubmodus kann die Bearbeitung eines Cursors wesentlich beschleunigt werden.

64-Bit-Ladevariante des SESAM/SQL-DBH

Die 64-Bit-Ladevariante des SESAM/SQL-DBH wird mit SESAM/SQL automatisch auf allen aktuellen BS2000-Servern geladen.
Zu erkennen ist dies in der DBH-Startmeldung am Insert „(64-Bit VERSION)“ für /390-Server und „(X86-64-VERSION)“ für x86-Server.

Erweiterte Pufferoptionen

Die 64-Bit-Ladevariante erlaubt eine performantere Abwicklung der Ein-/Ausgabelast durch einen größeren Maximalwert des Puffers für Systemzugriffsdaten (DBH-Option SYSTEM-DATA-BUFFER) und des Puffers für Anwenderdaten (DBH-Option USER-DATA-BUFFER), siehe Handbuch „Datenbankbetrieb“.

Gleichzeitig belasten die Puffer in diesem Fall nicht den normalen Adressraum der Task (2 GB), so dass für die anderen Optionen größere Werte möglich sind.

2.7 Ständige Verfügbarkeit und hohe Ausfallsicherheit

Bei OLTP-Anwendungen werden abhängig von der betrieblichen Organisation extrem hohe Anforderungen an die Verfügbarkeit gestellt:

- In Fehlersituationen müssen automatisch Backup-Systeme gestartet werden.
- Pflege- und Wartungsarbeiten müssen weitgehend im laufenden Betrieb möglich sein.

SESAM/SQL bietet zahlreiche Funktionen, um diesen Anforderungen gerecht zu werden.

Dynamische Rekonfiguration der DBH-Session

Dem SESAM/SQL-Systemverwalter stehen die DBH-Administrationskommandos RECONFIGURE-DBH-SESSION und RELOAD-DBH-SESSION zur Steigerung der Verfügbarkeit des independent DBH, zum Einspielen einer Korrekturversion des DBH während des DBH-Betriebs und zur dynamischen Rekonfiguration der DBH-Session zur Verfügung (siehe Handbuch „Datenbankbetrieb“).

Beide Administrationsanweisungen werden ohne Unterbrechung der DBH-Session ausgeführt. Aus Anwendersicht kommt es nicht zu einem DBH-Ausfall.

Automatische Erweiterung der Datenbankgrenzen

Wenn sich durch umfangreiche Einfügungen und Änderungen die vorher definierten physikalischen Grenzen der Datenbank als zu klein erweisen, erweitert SESAM/SQL die Grenzen automatisch während des Betriebs. Da die Erweiterung automatisch durchgeführt wird, erhöht sich die Verfügbarkeit erheblich.

Datendefinition und Utilities online

Der Datenbankverwalter kann seine Aufgaben wie Laden, Sichern und Nachfahren von SESAM-Sicherungsbeständen während des laufenden Betriebs durchführen; die Datenbank muss nicht abgeschaltet werden.
Ebenso können Datenbanken online aufgebaut werden. Die logische Datenbeschreibung (Schema) einer bestehenden Datenbank kann ebenfalls online geändert werden.
Situationen, in denen eine Datenbank abgeschaltet werden muss, sind äußerst selten.

Einsatz von Replikaten

Ein Replikat ist eine Datenbankkopie, die für die Wiedergewinnung oder auch als Schattendatenbank für die Recovery verwendet werden kann.

Durch die Verwendung von Replikaten als Schattendatenbanken können die Recovery-Zeiten kurz gehalten werden.

Verwendung von Fremdkopien

Mit SESAM/SQL-Server können Fremdkopien erzeugt werden, ohne die Datenbank aus dem laufenden Betrieb zu nehmen. Außerdem können Fremdkopien direkt, d.h. ohne den Umweg über ein Replikat, für die Recovery verwendet werden.

Verkürzung der Warmstartzeit

SESAM/SQL-Server ermöglicht den Beginn des OLTP-Betriebs bereits während des Warmstarts und bietet die Möglichkeit, die Warmstartzeit zu beeinflussen.

Space-Konzept

SESAM/SQL leitet aus der logischen Datenbeschreibung (Schema) automatisch den physikalischen Aufbau der Datenbank ab.

Mit der Storage Structure Language kann der Anwender bei Bedarf die Speicherorganisation beeinflussen und sie auf seinen Einsatzfall hin optimieren, zum Beispiel um bestimmte Zugriffe zu beschleunigen. Eine der Maßnahmen, die die Speicherstruktur beeinflusst, ist die physikalische Aufteilung der Datenbank auf bis zu 1000 Spaces (Dateien). Durch diese physikalische Aufteilung ist es in vielen Fällen nicht erforderlich, die ganze Datenbank abzuschalten, sondern nur die betroffenen Spaces. Zum Beispiel kann sich die Rekonstruktion defekter Teile oder die Sicherung auf einzelne Spaces beschränken.

Ein Space kann auf Pubsets mit „großen Dateien“ bis zu 4 TByte groß werden.
Sonst kann er bis zu 64 GByte groß werden.

2.8 Online Transaction Processing (OLTP)

OLTP-Anwendungen zeichnen sich dadurch aus, dass sehr viele Benutzer mit verschiedenen Aufgaben gleichzeitig und dialoggesteuert mit denselben Daten und Programmen arbeiten. Dies geschieht in der Regel unter der Steuerung eines Transaktionsmonitors. Typische Einsatzfälle für OLTP-Anwendungen sind Buchungssysteme, Auskunftssysteme oder Lagerhaltungssysteme.

OLTP-Anwendungen werden in BS2000 mit dem universellen Transaktionsmonitor openUTM realisiert. Der universelle Transaktionsmonitor openUTM ist optimal in das Betriebssystem integriert und bildet zusammen mit SESAM/SQL ein äußerst performantes OLTP-System. Informationen zu openUTM finden Sie in den UTM-Handbüchern, insbesondere im openUTM-Handbuch „Konzepte und Funktionen“.

Als eine Art Auftragsabwicklungszentrum koordiniert und verschickt openUTM die Aufträge, die von verschiedenen Datenstationen eingehen. SESAM/SQL stellt sicher, dass der gleichzeitige Zugriff vieler Endbenutzer auf die Datenbanken aus mehreren Verarbeitungsschritten heraus komplikationslos abläuft. Die Verarbeitung wird in logisch abgeschlossenen Einheiten, sogenannten Transaktionen, durchgeführt, die zwischen openUTM und SESAM/SQL synchronisiert werden. Transaktionen werden von openUTM und SESAM/SQL nur gemeinsam festgeschrieben oder zurückgesetzt. Der Benutzer am Bildschirm weiß in jeder Situation, an welcher Stelle der Verarbeitungsschritte er sich befindet. Die Datenbanken können zu jedem Zeitpunkt in einen konsistenten Zustand gebracht werden. Genaueres zur Synchronisation der openUTM- und SESAM/SQL-Transaktionen können Sie auf Seite 410 nachlesen.

Besonders eindrucksvoll dokumentieren openUTM und SESAM/SQL ihre Leistungsfähigkeit bei Großinstallationen, bei denen mehrere tausend Benutzer angeschlossen sind und in denen mehrere hundert Transaktionen pro Sekunde abgehandelt werden. In OLTP-Anwendungen übernimmt openUTM folgende wichtige Funktionen:

– openUTM sorgt dafür, dass auch bei sehr hohen Anwenderzahlen die Antwortzeiten kurz bleiben. So werden unter UTM-Steuerung zahlreiche Benutzeranforderungen durch wenige Systemtasks abgearbeitet. Dadurch wird der Verwaltungsaufwand des Betriebssystems minimiert.
– openUTM übernimmt die Überwachung, Steuerung und Abwicklung der Benutzeraufträge. openUTM ordnet die Aufträge den entsprechenden Programmen zu und steuert die Kommunikation mit dem Datenbanksystem.
– Die Bearbeitung erfolgt unter Transaktionssteuerung, d.h. eine Folge von zusammengehörenden Verarbeitungsschritten wird entweder ganz oder gar nicht ausgeführt.
– Bei Systemausfällen aktiviert openUTM einen automatischen und mit dem Datenbanksystem synchronisierten Wiederanlauf der Anwendung und der Bildschirmmaske.

SESAM/SQL und openUTM bieten damit eine ideale Basis für performante und sichere OLTP-Anwendungen.

2.9 Entwicklungswerkzeuge

Zur Entwicklung von SQL-Anwendungen für SESAM/SQL stehen zahlreiche Werkzeuge zur Verfügung.

Entwickler, die mit den Programmiersprachen der 3. Generation arbeiten, können mit ESQL-Produkten SQL-Anwendungen unter COBOL erstellen. Für C-Anwendungen in BS2000 müssen die SQL-Aufrufe in einem ESQL-COBOL-Modul gebündelt werden, der aus C heraus aufgerufen wird.

DRIVE ist eine Programmiersprache der 4. Generation. Sie bietet hohen Komfort und mächtige Sprachmittel, wodurch sich im Entwicklungsprozess erhebliche Produktivitätsgewinne erzielen lassen.

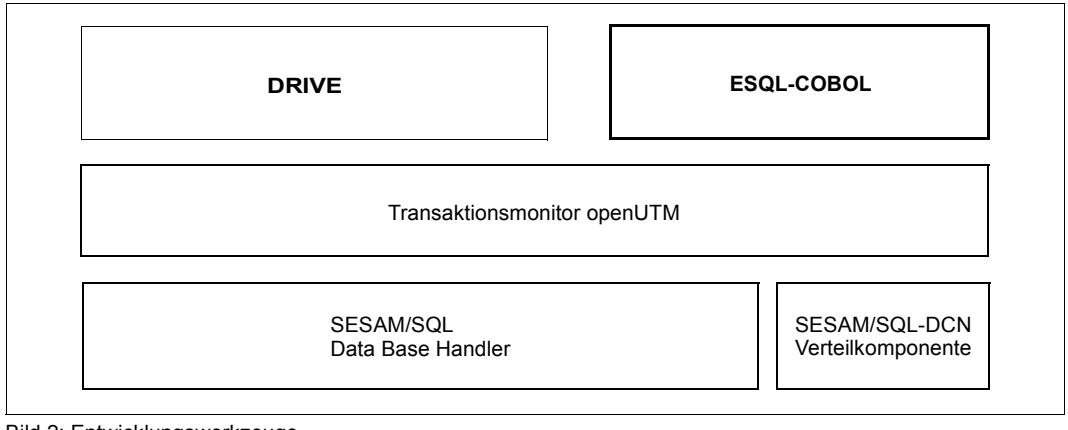


Bild 2: Entwicklungswerkzeuge

2.9.1 DRIVE

Zur Entwicklung von SESAM/SQL-Anwendungen bietet sich mit DRIVE (BS2000) eine komfortable Programmiersprache der 4. Generation an. DRIVE eignet sich zur Entwicklung von OLTP-Anwendungen unter openUTM, von Client/Server-Anwendungen auf BS2000-Systemen sowie zur Entwicklung von Programmen zur individuellen Datenverarbeitung (IDV).
Der DRIVE-Compiler wird eingesetzt, wenn besonders hohe Anforderungen an den Durchsatz gestellt werden.

In Client-Server-Architekturen unterstützt DRIVE die transaktionsgesicherte verteilte Verarbeitung mit openUTM-D.

Die genaue Beschreibung der DRIVE-Funktionen entnehmen Sie bitte den Handbüchern zu DRIVE bzw. DRIVE/WINDOWS.

2.9.2 ESQL-COBOL

Der Precompiler ESQL-COBOL (BS2000) ermöglicht die Ausführung von COBOL-Programmen, in die SQL-Anweisungen eingebettet sind. Der Anfang und das Ende einer SQL-Anweisung werden jeweils markiert. Der Precompiler kann damit die SQL-Anweisungen erkennen und erzeugt ein SQL-Bindelademodul (Link-and-Load-Module, LLM), in dem die SQL-Anweisungen zusammengefasst sind. Das COBOL-Programm, das keine SQL-Anweisungen mehr enthält, wird mit dem COBOL-Compiler übersetzt. Die folgende Abbildung zeigt den geschilderten Ablauf.

Programme mit National-Datentypen können nur mit entsprechender BS2000-Systemumgebung (vor-)übersetzt werden, siehe Seite 28.

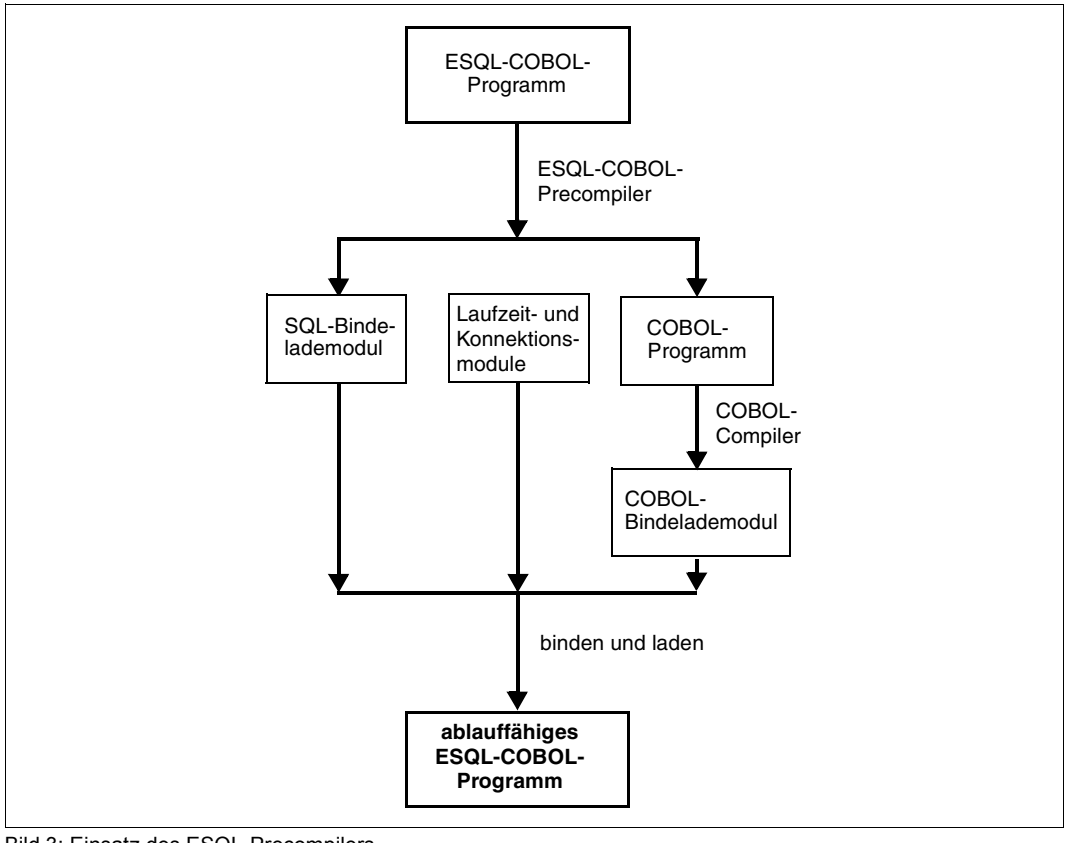


Bild 3: Einsatz des ESQL-Precompilers

Über Precompiler-Optionen kann der Anwender die Eigenschaften der Precompilierung festlegen. So ist u.a. einstellbar, dass nur standardkonforme SQL zugelassen wird, um dadurch portierbare SQL-Programme zu erzeugen.

Mit Hilfe der sogenannten dynamischen SQL ist es möglich, SQL-Anweisungen erst während des Ablaufs eines ESQL-Programms anzugeben. Damit sind z.B. Dialoganwendungen mit variabel gestalteten Abfragen und Änderungen der Datenbank realisierbar.

Ein ESQL-Programm kann auch CALL-DML-Anweisungen enthalten. SQL-Anweisungen und CALL-DML-Anweisungen können auch innerhalb einer Transaktion gemeinsam enthalten sein. Dazu werden SQL-Anweisungen in eine CALL-DML-Transaktion eingebettet.

Die genaue Beschreibung des ESQL-Precompiler entnehmen Sie bitte dem Handbuch „ESQL-COBOL für SESAM/SQL-Server“.

Für SESAM/SQL-Server wird kein ESQL-C-Precompiler angeboten. Stattdessen gibt es die Empfehlung, die SQL-Anweisungen aus einem C-Programm zu extrahieren und in ein ESQL-COBOL-Programm einzuschalen. Aus dem C-Programm heraus können dann die COBOL-Prozeduren mit den jeweiligen SQL-Anweisungen aufgerufen werden. Die Werte für SQL-Hostvariablen werden als Parameter an die COBOL-Prozedur übergeben.

2.10 SESAM/SQL im Client-Server-Umfeld

Über Client-Server-Architekturen kann die Mächtigkeit des Data Centers mit den komfortablen grafischen Bedienoberflächen und der Vielzahl von Standard-Anwendungen auf PCs und Workstations zu einem optimalen Gesamtsystem verbunden werden.

Client-Server-Lösungen sind in verschiedenen Grundformen denkbar, je nachdem welche Dienste der Client vom Data Center in Anspruch nimmt und welche Aufgaben er selbst übernimmt.

2.10.1 Grundformen von Client/Server-Architekturen

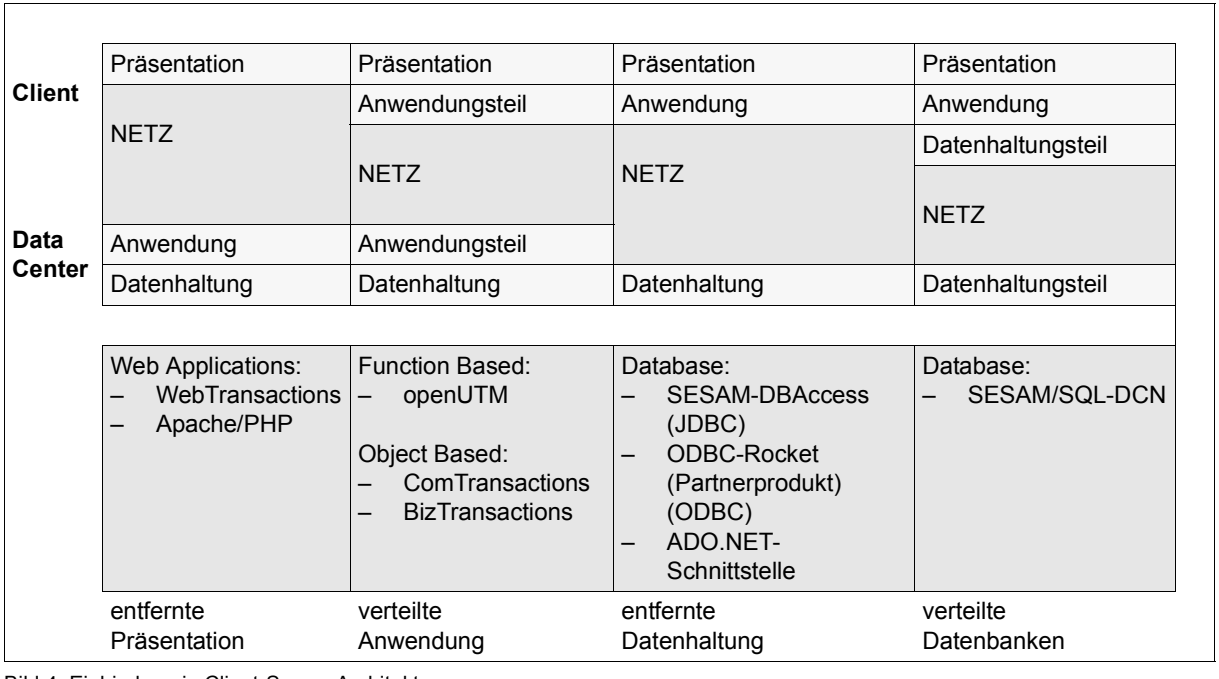


Bild 4: Einbindung in Client-Server-Architekturen

Entfernte Präsentation

Bei Anwendungen mit grafischer Bedienoberfläche wird ein bedeutender Teil der Rechenleistung für die Aufbereitung der Präsentation eingesetzt. Durch die Auslagerung auf ein Client-System wird das Data Center entlastet und damit frei für weitere Aufgaben. Auf diese Weise wird die Qualität eines intelligenten grafischen Arbeitsplatzes mit der eines Data Centers kombiniert. Über ein einheitliches „Look and Feel“ greift der Benutzer auf lokale und entfernte Anwendungen zu. Der Nutzen dieser Client-Server-Variante ist die Erhöhung der Produktivität am Arbeitsplatz bei gleichzeitiger Entlastung des Data Centers.

Verteilte Anwendung

Bei der verteilten Anwendung werden die einzelnen Aufgaben einer Anwendung auf die Systeme verteilt, die sich am besten dafür eignen. Durch diese Verteilung kann insgesamt eine größere Rechenleistung genutzt werden.

Entfernte Datenhaltung

Bei der entfernten Datenhaltung arbeitet die Client-Anwendung mit einer Datenbank auf einem entfernten Data Center. Dies ermöglicht eine zentrale und damit widerspruchsfreie (konsistente) sowie sichere Datenhaltung. Gleichzeitig kann das Data Center die volle Leistung für den Datenzugriff nutzen, da es weder die Anwendung noch die Präsentation steuern muss.

Einen Spezialfall der entfernten Datenhaltung stellen die verteilten Datenbanken dar.

Verteilte Datenbanken

Verteilte Datenbanken ermöglichen dem Anwender, die Datenorganisation auf die betriebliche Organisationsstruktur auszurichten, z.B. Zentrale und Filialen. Die Daten werden auf den Servern abgelegt, an denen sie vorrangig benötigt werden. Damit lässt sich der Engpass Netz minimieren. Dies gilt insbesondere dann, wenn öffentliche Netze benutzt werden, die in der Regel geringere Übertragungsraten haben. Gleichzeitig werden so die Übertragungskosten gering gehalten.

In den Client-Server-Architekturen „Entfernte Datenhaltung“ und „Verteilte Datenbanken“ stellt SESAM/SQL Zusatzkomponenten zur Verfügung.

2.10.2 Entfernte Datenhaltung mit Zugriff über Standard-Schnittstellen

Über Zusatzprodukte wie SESAM-DBAccess (JDBC-Schnittstelle), ODBC-Rocket (Partnerprodukt mit ODBC-Schnittstelle) oder die ADO.NET-Schnittstelle können Windows-Systeme auf SESAM/SQL-Datenbanken zugreifen, sie verarbeiten oder auch verändern. Beispielsweise können Daten in eine Tabellenkalkulation eingelesen werden, oder Adressen, die sich in der Datenbank befinden, für die Serienbrieferstellung einer Textverarbeitung verwendet werden.

Die Server- und Client-Komponente von SESAM-DBAccess ist im Lieferumfang von SESAM/SQL enthalten und muss nicht extra bestellt werden.

Remote-Zugriff vom PC mit ODBC

SESAM/SQL-Server unterstützt die von Microsoft definierte ODBC-Schnittstelle (Open Database Connectivity) zur Kommunikation von Windows-Anwendungen mit Datenbanksystemen. Die mit SESAM/SQL-Server ausgelieferte CD-ROM enthält eine Demo-Version des Partnerprodukts ODBC-Rocket. ODBC-Rocket der Firma gfs in Hamburg bietet eine ODBC-Schnittstelle für SESAM/SQL.

Anwendungsbereiche für den Datenbankzugriff aus Windows-Anwendungen auf zentrale Datenbestände sind alle Client-Server-Applikationen, bei denen die Anwendung auf dem Windows-System liegt und die zu verarbeitenden Daten auf einer SESAM/SQL-Datenbank geführt werden:

- Standard-Windows-Anwendungen
  wie ACCESS, EXCEL oder WORD
- Individuell erstellte Windows-Anwendungen (z.B. mit Visual Basic, Visual C++).
  Damit können Sie zum Beispiel:
  – individuelle Auswertungen durchführen
  – Daten in eine Tabellenkalkulation einlesen
  – für Serienbrieferstellung Adressen in eine Textverarbeitung einlesen

  Ein weiteres Beispiel ist die Erstellung von Geschäftsgrafiken aus Daten, die auf dem Datenserver liegen. Eine solche Anwendung kann der erste Schritt zu einem Data Warehouse sein.

Remote-Zugriff mit ADO.NET

SESAM/SQL-Server unterstützt die von Microsoft definierte ADO.NET-Schnittstelle (ActiveX Data Objects) zur Kommunikation von Windows-Systemen mit Datenbanksystemen im Client-Server-Umfeld.

Über den ADO.NET-Treiber, der zusammen mit SESAM/SQL-Server ausgeliefert wird, können Sie datenbank-unabhängige Zugriffe auf SESAM/SQL aus dem .NET-Umfeld in Windows realisieren.

Webserver mit Anschluss an SESAM/SQL

Für FUJITSU Server BS2000 SE Serie wird auf einer Application Unit unter LINUX ein PDO-Treiber für die PHP-Schnittstelle auf einen Apache Webserver angeboten.

Über diesen PDO-Treiber können Zugriffe auf SESAM/SQL-Datenbanken in PHP-Webseiten und -Skripts durchgeführt werden.

Der PHP-Code wird ausschließlich auf der Server Unit ausgeführt. Er bleibt dem Web-Anwender verborgen. Nur der HTML-Code wird zum Client übertragen. Dies erfüllt auch erhöhte Sicherheitsanforderungen.

Neben diesem neuen PDO-Treiber wird wie bisher die PHP-Zugriffsfunktionalität über einen Apache Webserver unter POSIX in BS2000 angeboten.

Mit bestehenden SESAM-Anwendungen in das Internet/Intranet: WebTransactions

Bestehende SESAM/UTM-Anwendungen können mit WebTransactions auf komfortable Weise webfähig gemacht werden. Dies gilt speziell auch für Anwendungen, die mit DRIVE erstellt wurden.

WebTransactions ist ein bereits sehr erfolgreich am Markt eingeführtes Produkt mit einem breiten Spektrum von Einsatzmöglichkeiten bei der Integration von geschäftsrelevanten Anwendungen und deren Daten in das World Wide Web.

Mit WebTransactions kann Folgendes erreicht werden:

- Host-Anwendungen und -Daten können unverändert in das Internet, Intranet oder Extranet integriert werden
- die Bildschirmmasken der Host-Anwendung können verbessert und verschönert werden
- die Host-Anwendung kann durch Zusammenfassen bzw. Erweitern von Dialogschritten bedarfsgerecht verändert werden (Dialog-Reengineering)
- es können sogar mehrere verschiedene Anwendungen unter einer Web-Oberfläche zusammengeführt und parallel betrieben werden

Gegenwärtig werden beliebige Mainframe-Anwendungen auf BS2000- und MVS-Systemen sowie OLTP-Anwendungen mit openUTM auf BS2000, UNIX-Systemen und Windows unterstützt.

Java: JDBC-Schnittstelle von SESAM/SQL-Server

Das Standard-Call-Level-Interface beim Zugriff auf SQL-Datenbanken für Java-Anwendungen, Java Server Pages, Java Servlets und Java Applets ist JDBC (Java Database Connectivity). Über die JDBC-Schnittstelle können datenbankunabhängige Java-Zugriffe mit SESAM/SQL-Server ablaufen. Somit können auch externe Java-Anwendungen mit SESAM/SQL-Server zusammenarbeiten.

SESAM/SQL-Server bietet die JDBC-Funktionalität gemäss JDBC-Standard V4.0. Die entsprechenden Treiber sind in SESAM/SQL-Server enthalten und werden auf einem Datenträger zusammen mit SESAM/SQL-Server ausgeliefert. Sie sind Treiber vom Typ 4, d.h. „native-protocol fully Java technology-enabled driver“. Dies hat gegenüber anderen Typen wie „ODBC/JDBC bridges“ und „partly Java technology-enabled drivers“ den Vorteil, dass keinerlei Binärcode auf der Client-Maschine installiert werden muss.

2.10.3 Verteilte Datenbanken mit SESAM/SQL-DCN

Um SESAM/SQL-Datenbanken, die auf unterschiedlichen BS2000-Rechnern liegen, mit einem Anwenderprogramm zu bearbeiten, steht die Verteilkomponente von SESAM/SQL, das Produkt SESAM/SQL-DCN, zur Verfügung. Bei der Bearbeitung dieser sogenannten verteilten Datenbanken besteht für das Anwenderprogramm kein Unterschied, ob die Daten auf dem lokalen oder auf einem entfernten Rechner liegen.
Im Anwenderprogramm befinden sich keine Informationen über den Ort der Daten, d.h. die Verteilung ist für die Anwendung transparent. Die gesamte Kommunikation wird für den Anwender unsichtbar abgewickelt. Anwenderprogramme können also unverändert auf einem beliebigen am Verbund beteiligten Rechner eingesetzt werden.

Da SESAM/SQL-DCN eine Änderungstransaktion, an der Datenbanken verschiedener Rechner beteiligt sind, mit einem Zwei-Phasen-Ende-Protokoll (Two-Phase-Commit) abschließt, ist die Konsistenz der Daten auch netzweit gesichert. SESAM/SQL garantiert also den konsistenten Datenbestand im Netz auch bei Datenänderungen auf mehreren Rechnern. Deadlock- und Longlock-Situationen werden über Rechnergrenzen hinweg erkannt und aufgelöst.

Je nach Anwendungsfall ergeben sich folgende Vorteile beim Einsatz von SESAM/SQL-DCN:

- Höhere Leistungsfähigkeit
  Durch das Verarbeiten der Anwenderaufträge auf verschiedenen Rechnern kann eine Durchsatzsteigerung erzielt werden.
- Höhere Verfügbarkeit
  Der Ausfall eines Rechners bewirkt nicht den Ausfall des Gesamtsystems.
- Flexible Organisation
  Die Arbeitsabläufe müssen sich nicht an einem zentralen Rechenzentrum orientieren.

Detailliertere Ausführungen zur Arbeitsweise von SESAM/SQL-DCN finden Sie im Abschnitt „Verteilte Verarbeitung mit SESAM/SQL-DCN“ auf Seite 283.

2.11 Weitere datenbanknahe Produkte und Anwendungen

SESAM-KLDS - kompatible Schnittstelle für lineare Datenstrukturen

Vor allem in der öffentlichen Verwaltung setzen Anwender Produkte verschiedener Hersteller ein. Daraus ergibt sich die Notwendigkeit, dass bestimmte Programme auf allen Anlagen ablaufen müssen.

Mit dem Produkt SESAM-KLDS können systemneutrale Programme für die Verarbeitung linearer Datenstrukturen erstellt werden. Die Datenbankaufrufe erfolgen über die Datenbank-Schnittstelle KLDS, die unter der Federführung des Bundesministeriums für Inneres definiert wurde.
SESAM-KLDS setzt Datenbankaufrufe im KLDS-Format in CALL-DML-Anweisungen um. SESAM-KLDS-Anwenderprogramme sind auch in openUTM-Umgebung möglich.

Alle Funktionen der KLDS-Schnittstelle sind in einem eigenen Handbuch beschrieben, siehe Handbuch „SESAM-KLDS (BS2000)“.

Entfernte Ausgabe von Informationen mit SNMP

SNMP steht für **S**imple **N**etwork **M**anagement **P**rotocol und wurde als Protokoll für Netzmanagement-Dienste in TCP/IP-Netzwerken entwickelt. Inzwischen hat sich der Anwendungsbereich von SNMP um System- und Anwendungsmanagement bis hin zum Management von Middleware-Produkten wie Datenbanken und Transaktionsmonitoren erweitert. Ähnlich wie bei TCP/IP steht auch der Name SNMP nicht nur für das Protokoll allein, sondern für das gesamte auf SNMP basierte Management-System. SNMP folgt einer Client/Server-Architektur mit der Management-Plattform als Client und den Management-Agenten als Server.

Für das SNMP-Management von SESAM/SQL in BS2000 gibt es zwei Subagenten:

- Subagent zum Management von SESAM-Datenbanken, enthalten in der SNMP Standard Collection BS2000 (Softwareprodukt SSC-BS2). Er liefert Informationen über SESAM-Datenbanken und SESAM-DBHs. Dieser Subagent ist ausführlich im Handbuch „SNMP Management für BS2000“ beschrieben.
- Subagent zur Ausgabe von Daten des Performance-Monitors SESMON.
  Dieser Subagent wird auch als **SESAM-MON-Subagent** bezeichnet. Er ist ausführlich im Handbuch „Datenbankbetrieb“, Kapitel „Betriebsdaten ausgeben mit SESMON“, beschrieben.

3 Beispieldatenbank

Zum Lieferumfang von SESAM/SQL-Server gehört die Beispieldatenbank AUFTRAGKUNDEN. Die Struktur dieser SESAM/SQL-Datenbank wird beschrieben im Abschnitt „Aufbau der Beispieldatenbank“ auf Seite 59.

Auf die Datenbank AUFTRAGKUNDEN beziehen sich die Beispiele in den Handbüchern zu SESAM/SQL-SERVER.

Die aufgebaute Datenbank finden Sie in der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB.

Die Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB enthält alle Bestandteile, die Sie benötigen, um die in den Handbüchern beschriebenen Beispiele selber auszuprobieren und eigene Anwendungen in einer überschaubaren Umgebung zu entwickeln.

Die Bestandteile der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB sind:

- Readme-Datei mit einer Übersicht über alle enthaltenen Dateien und einer detailierten Anleitung zum Umgang mit der Beispieldatenbank
- Startprozeduren zum Starten der benötigten SESAM/SQL-Programme
- Dateien der aufgebauten Datenbank AUFTRAGKUNDEN
- Anweisungsdateien und ESQL-COBOL-Programme mit Ablaufbeispielen für wichtige Datenbank-Anweisungen

3.1 Beispieldatenbank starten

Zum Starten der Beispieldatenbank gehen Sie wie folgt vor:

1. Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB bereitstellen und an die Ablaufumgebung anpassen:

   Während einige Bestandteile der Beispieldatenbank als Bibliothekselemente angesprochen werden können, müssen die Spaces der Datenbank AUFTRAGKUNDEN, Konfigurationsdateien und Ladedateien mit Anwenderdaten in eine Arbeitskennung kopiert werden.

   Desweiteren sind einige Parameter in den mitgelieferten Prozeduren, wie zum Beispiel die Kennungen der Systemdateien, abhängig von der jeweiligen Ablaufumgebung.

   In der Readme-Datei der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB finden Sie eine detailierte Beschreibung für alle Schritte, die vor dem 1. Starten notwendig sind.

2. Starten des DBH:

   Der Data Base Handler (DBH) ist die Komponente von SESAM/SQL, die alle Datenbankzugriffe einer DBH-Session analysiert, ausführt und koordiniert.

   Verwenden Sie zum Starten des DBH die mitgelieferte Prozedur in der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB.

   Weitere Informationen zum Umgang mit dem DBH finden Sie im Handbuch „Datenbankbetrieb“.

3. Starten des SESADM (optional):

   Zur Administration des DBH können Sie das Administrationsprogramm SESADM starten.

   Verwenden Sie dazu die mitgelieferte Prozedur in der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB.

   Weitere Informationen zum Umgang mit dem Administrationsprogramm SESADM finden Sie im Handbuch „Datenbankbetrieb“.

4. Starten des Utility-Monitors:

   Über die menügesteuerte Oberfläche des Utility-Monitors können Sie eine Datenbank verwalten.

   Verwenden Sie zum Starten des Utility-Monitors die mitgelieferte Prozedur in der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB.

   Weitere Informationen zum Umgang mit dem Utility-Monitor finden Sie im Handbuch „Utility-Monitor“.

Nun können Sie über die Menüoberfläche des Utility-Monitors auf die Datenbank zugreifen.

3.2 Beispieldatenbank nutzen

Mit der Beispieldatenbank AUFTRAGKUNDEN können Sie einzelne Handbuchbeispiele zur Syntax von SQL-Anweisungen ausprobieren. Sie können auf diesen Beispielen aufbauen, um eigene SQL-Anweisungen entwickeln.

Für die Beispieldatenbank wurden ausgewählte SQL-Anweisungen aus den Handbüchern zu ablauffähigen Einheiten zusammengestellt. Die Anweisungsdateien und ESQL-COBOL-Programme zeigen Anwendungsmöglichkeiten der SQL-Anweisungen auf und sollen zur Entwicklung eigener Programme anregen.
Sie sind in der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB enthalten.

3.2.1 Einzelne SQL-Anweisungen

Zum Ausprobieren der Beispieldatenbank AUFTRAGKUNDEN eignen sich die meisten Beispiele im Kapitel „SQL-Funktionsumfang von SESAM/SQL“ auf Seite 77 sowie in den Handbüchern „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“ und „SQL-Sprachbeschreibung Teil 2: Utilities“.

Rufen Sie dazu im Utility-Monitor die Maske SQL - SQL-STATEMENTS auf und geben Sie dort die gewünschte SQL-Anweisung ein.

In der Maske SQL können Sie beliebige, dynamisch übersetzbare SQL-Anweisungen eingeben und in einer Datei protokollieren lassen. Es bietet sich daher an, hier neue SQL-Anweisungen zu entwickeln und diese anschließend aus der Protokolldatei syntaktisch richtig in einen komplexeren Zusammenhang zu übernehmen.

Weitere Informationen zum Umgang mit dem Utility-Monitor finden Sie im Handbuch „Utility-Monitor“.

3.2.2 Anweisungsdateien und ESQL-COBOL-Programme

Ein wesentlicher Bestandteil der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB sind Anweisungsdateien und ESQL-COBOL-Programme. In ihnen werden Handbuchbeispiele themenbezogen zu ablauffähigen Beispielen zusammengefasst.

Beispiele werden durch nebenstehendes Symbol gekennzeichnet, wenn sie als Bestandteil einer Anweisungsdatei oder eines ESQL-COBOL-Programms in der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB enthalten sind.

Die Anweisungsdateien und ESQL-COBOL-Programme umfassen folgende Themen:

Thema	**Ablaufumgebung**
Datenbank AUFTRAGKUNDEN neu aufbauen	Anweisungsdatei
Anwenderdaten laden in die Datenbank AUFTRAGKUNDEN	Anweisungsdatei
Anwenderdaten entladen in unterschiedlichen Datenformaten	Anweisungsdatei
Schwerpunkt DDL-Anweisungen	Anweisungsdatei
Schwerpunkt DML-Anweisungen	Anweisungsdatei
Schwerpunkt Utility-Anweisungen	Anweisungsdatei
Schwerpunkt Neuerungen in SESAM/SQL-Server V3.2, V4.0, V5.0, V6.0, V7.0, V8.0 und V9.0	Anweisungsdateien
Schwerpunkt partitionierte Tabellen	Anweisungsdatei
Schwerpunkt Unicode-Tabellen	Anweisungsdatei
Schwerpunkt Prozeduren und User Defined Functions (UDF)	Anweisungsdateien
Auswahl von Sätzen aus einer Tabelle	ESQL-COBOL-Programm
Einfügen von Sätzen in eine Tabelle	ESQL-COBOL-Programm
Ändern von Sätzen einer Tabelle	ESQL-COBOL-Programm
Löschen von Sätzen aus einer Tabelle	ESQL-COBOL-Programm
Verwendung von dynamischer SQL	ESQL-COBOL-Programm
Arbeiten mit BLOB-Objekten	ESQL-COBOL-Programm
Nutzung von Prozeduren und User Defined Functions (UDF)	ESQL-COBOL-Programm

Zum Starten der Anweisungsdateien und ESQL-Programme gehen Sie wie folgt vor:

- Anweisungsdateien:
  Rufen Sie im Utility-Monitor die Maske IFP - INSTRUCTION FILE PROCESSING auf und geben Sie den Namen der Anweisungsdatei ein.
- ESQL-COBOL-Programmdateien:
  Starten Sie die ESQL-COBOL-Programme über die mitgelieferte Prozedur. Neben den ablauffähigen Programmen sind in der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB die Quellprogramme und die Prozeduren enthalten, die Sie zum Übersetzen und Binden eines Quellprogramms benötigen.

In der Readme-Datei der Bibliothek SIPANY.SESAM-SQL.<ver>.MAN-DB finden Sie detaillierte Beschreibungen für alle durchzuführenden Schritte.

Neue Anweisungsdateien und ESQL-COBOL-Programme können Sie im Editor EDT entwickeln und analog zu den bereits vorhandenen Dateien als Bibliothekselement speichern.

3.3 Aufbau der Beispieldatenbank

Im Folgenden wird der physikalische und logische Aufbau der Beispieldatenbank AUFTRAGKUNDEN erläutert. Hinweise auf weitere Informationen zu den verwendeten Begriffen und Einstellungen finden Sie jeweils am Ende eines Abschnitts.

3.3.1 Storage Groups

Eine Storage Group fasst Speichermedien einer BS2000-Katalogkennung unter einem Namen zusammen.

Die Spaces der Datenbank AUFTRAGKUNDEN werden auf drei Storage Groups verteilt:

- Storage Group STOGROUP1
  für den Catalog Space und die Anwender-Spaces sowie die DA-LOG-Dateien.
- Storage Group STOGROUP2
  für die CAT-REC- und die CAT-LOG-Datei
- Storage Group STOGROUP3
  für die Sicherungskopien des Catalog Space und der Anwender-Spaces.

In der mitgelieferten Version der Beispieldatenbank AUFTRAGKUNDEN werden alle drei Storage Groups im Default Pubset der BS2000-Kennung angelegt. Auf diese Weise kann auf die Datenbank zugegriffen werden unabhängig von der jeweiligen Ablaufumgebung.

Im Produktiveinsatz sollte die CAT-REC-Datei aber auf einem anderen Speichermedium angelegt werden als der Catalog Space und die Anwender-Spaces, um auch bei Verlust eines Speichermediums ein Media-Recovery durchführen zu können.

Weitere Informationen zum Aufbau einer SESAM/SQL-Datenbank und zum Media-Recovery finden Sie in dem Abschnitt „SQL-Objekte einer SESAM/SQL-Datenbank“ auf Seite 79 und in dem Abschnitt „Media-Recovery“ auf Seite 211.

3.3.2 Anwender-Spaces

Die Anwenderdaten der Datenbank AUFTRAGKUNDEN werden auf folgende drei Anwender-Spaces verteilt:

- Space TABLESPACE zum Speichern von Tabellen
- Space INDEXSPACE zum Speichern von Indizes
- Space BLOBSPACE zum Speichern von BLOB-Objekten

Werden Tabellen und die dazugehörigen Indizes auf getrennten Spaces angelegt, können die Möglichkeiten des Media-Recovery effizient und differenziert ausgenutzt werden.

Bei größeren Datenbanken sollten Sie jede Tabelle auf einem eigenen Space anlegen, um die Zugriffsmöglichkeiten auf einzelne Tabellen besser steuern zu können und die Ausfallzeiten durch Recoverymaßnahmen gering zu halten.

Weitere Informationen zum Media-Recovery finden Sie im Abschnitt „Media-Recovery“ auf Seite 211.

3.3.3 Schemata

Die Beispieldatenbank AUFTRAGKUNDEN ist in folgende Schemata unterteilt:

- Schema AUFTRAGSVER zur Verwaltung von Aufträgen,
- Schema TEILE zur Verwaltung von Bauteilen,
- Schema ZUSAETZE zur Verwaltung von BLOB-Objekten.

Um die Tabellen der Datenbank hier vollständig zu beschreiben, wird neben den Daten einer Tabelle jeweils auch die Definition der Tabelle dargestellt. Die Definition einer Tabelle mit CREATE TABLE setzt Begriffe wie Namen, Datentyp, Integritätsbedingungen voraus. Diese Begriffe werden im Abschnitt „SQL-Objekte einer SESAM/SQL-Datenbank“ auf Seite 79 beschrieben.

NULL-Werte in den Tabellen sind als leere Felder dargestellt.

3.3.3.1 Schema AUFTRAGSVER

Im Schema AUFTRAGSVER der Datenbank AUFTRAGKUNDEN ist ein Informationssystem zur Auftragsabwicklung eines kleinen Data Centers realisiert. Es sollen Informationen über Kunden, über Kontaktpersonen bei den Kunden, über Aufträge dieser Kunden und über Leistungen erfasst werden, die zu diesen Aufträgen erbracht wurden. Das Schema AUFTRAGSVER enthält die Tabellen KUNDE, KONTAKT, AUFTRAG, LEISTUNG und AUFSTAT.

Folgendes Bild zeigt in der Übersicht die Basistabellen des Schemas AUFTRAGSVER und die Abhängigkeiten zwischen den Tabellen.

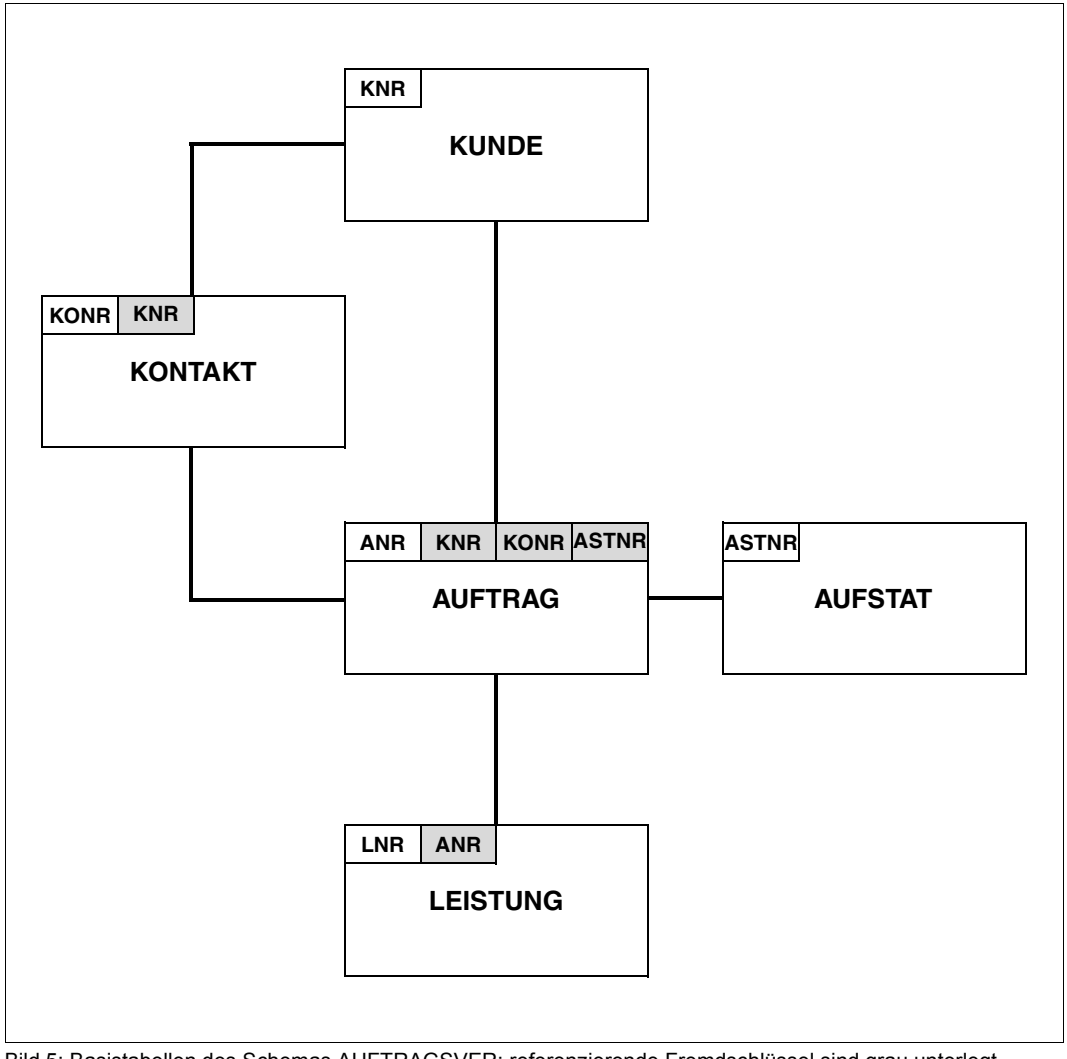


Bild 5: Basistabellen des Schemas AUFTRAGSVER; referenzierende Fremdschlüssel sind grau unterlegt

Tabelle KUNDE

Die Tabelle KUNDE enthält Informationen über die Kunden. Neben einer eindeutigen Kundennummer enthält die Tabelle den Namen, die Adresse, die Telefonnummer sowie Informationen über die Branche des Kunden. Die Tabelle KUNDE wird mit der folgenden Tabellendefinition erzeugt:

```
CREATE TABLE          kunde
(knr                  INTEGER CONSTRAINT knr_primary PRIMARY KEY,
 firma                CHAR(40) CONSTRAINT firma_notnull NOT NULL,
 strasse              CHAR(40),
 plz                  NUMERIC(5),
 ort                  CHAR(40),
 land                 CHAR(3),
 ktelefon             CHAR(25),
 kinfo                CHAR(50)
 CONSTRAINT PlausPlz  CHECK( land IS NULL OR plz IS NULL OR
                      (land = 'D' AND plz >= 00000)
                      OR (land <> 'D'))
)
```

KNR ist der Primärschlüssel von KUNDE. Es ist eine CHECK-Bedingung definiert, die die Plausibilität der Postleitzahlen abprüft. Die Tabelle KUNDE enthält folgende Daten:

knr	firma	strasse	plz	ort	land	ktelefon	kinfo
100	Siemens AG	Otto-Hahn-Ring 6	81739	Muenchen	D	089/636-8	Elektro
101	Login GmbH	Rosenheimer Str. 34	81667	Muenchen	D	089/4488870	PC Netzwerke
102	JIKO GmbH	Posener Str. 12	30659	Hannover	D	0551/123874	Import-Export
103	Plenzer Trading	Paul-Heyse-Str. 12	80336	Muenchen	D	089/923764	Fruechtehandel
104	Jonas Fischladen	Hirschgartenstr. 12	12587	Berlin	D	016/5739921	Einzelhandel
105	Pudelshop Anke	Am Muehlentor 26	41179	Moenchengladbach	D	040/873562	Dienstleistung
106	Foreign Ltd.	26 West York St.		New York, NY	USA	001703/2386 532	Handelsvertretung
107	Externa & Co KG	Berner Weg 78	3000	Bern 33	CH		Anwaltskanzlei

Tabelle 4: Daten der Tabelle KUNDE

Tabelle KONTAKT

Die Tabelle KONTAKT enthält Informationen über die Kontaktpersonen bei einem bestimmten Kunden. Sie besteht aus einer eindeutigen Kontaktnummer, der Nummer des Kunden in der Tabelle KUNDE, Vorname, Nachname, Anrede und Telefonnummer der Kontaktperson sowie Informationen über Stellung und Abteilung der Kontaktperson und den Anlass des Kontakts. Die Tabelle KONTAKT wird mit der folgenden Tabellendefinition erzeugt:

```
CREATE TABLE     kontakt
(konr            INTEGER CONSTRAINT konr_primary PRIMARY KEY,
 knr             INTEGER CONSTRAINT ko_knr_notnull NOT NULL,
 vorname         CHAR(25),
 nachname        CHAR(25) CONSTRAINT name_notnull NOT NULL,
 anrede          CHAR(20),
 kotelefon       CHAR(25),
 funktion        CHAR(50),
 abteilung       CHAR(30),
 koinfo          CHAR(50),
 CONSTRAINT      ko_knr_ref_kunde FOREIGN KEY (knr) REFERENCES kunde
)
```

KONR ist der Primärschlüssel der Tabelle KONTAKT. Für die Tabelle ist eine Referenzbedingung definiert. Der Fremdschlüssel KNR bezieht sich auf den Primärschlüssel KUNDE.KNR der referenzierten Tabelle KUNDE.
Die Tabelle KONTAKT enthält folgende Daten:

konr	knr	vorname	nachname	anrede	kotelefon	funktion	abteilung	koinfo
10	100	Walter	Kuehne	Herr Dr.	089/6361896	Vorstand	Personal	
11	100	Stefan	Walkers	Herr	089/63640182	Sekretaer	Vertrieb	
20	101	Roland	Loetzerich	Herr	089/4488870	Geschaefts-fuehrer		Netz-werke
25	102	Ewald	Schmidt	Herr	0551/123873	Schulung		
26	103	Beate	Kredler	Frau	089/923764	Organisation		SQL-Kurs
30	104	Xaver	Bauer	Herr	016/6739921	Verkaeufer		
35	105	Anke	Buschmann	Frau	02161/584097	Geschaefts-fuehrer		
40	106	Mary	Davis	Ms.	001703/2386531	Leitung	Einkauf	
41	106	Robert	Heinlein	Mr.	001703/2386532	Ausbilder	Einkauf	

Tabelle 5: Daten der Tabelle KONTAKT

Tabelle AUFSTAT

Die Tabelle AUFSTAT ermöglicht die Zuordnung der Auftragsstatus-Nummern in der Tabelle AUFTRAG (Spalte ASTNR) zu den entsprechenden Texten. Sie enthält die eindeutige Auftragsstatus-Nummer und den entsprechenden Text.

Die Tabelle AUFSTAT wird mit der folgenden Tabellendefinition erzeugt:

```
CREATE TABLE     aufstat
(astnr           INTEGER CONSTRAINT astnr_primary PRIMARY KEY,
 astxt           CHAR(15) CONSTRAINT astxt_notnull NOT NULL
)
```

Die Tabelle AUFSTAT enthält folgende Daten:

astnr	**astxt**
1	geplant
2	Vertrag
3	erledigt
4	abgerechnet
5	Ablage

Tabelle 6: Daten der Tabelle AUFSTAT

Tabelle AUFTRAG

In der Tabelle AUFTRAG befinden sich die Grunddaten zu einem Auftrag. Die Tabelle enthält eine eindeutige Auftragsnummer, Verweise auf den Kunden und den Kundenkontakt, das Datum der Auftragserteilung, die Bezeichnung des Auftrags, Ist- und Soll-Termin der Fertigstellung sowie eine Auftragsstatus-Nummer.
Die Tabelle AUFTRAG wird mit der folgenden Tabellendefinition erzeugt:

```
CREATE TABLE  auftrag
(anr          INTEGER CONSTRAINT anr_primary PRIMARY KEY,
 knr          INTEGER CONSTRAINT a_knr_notnull NOT NULL,
 konr         INTEGER,
 adatum       DATE DEFAULT CURRENT_DATE,
 atext        CHAR(30),
 fertigist    DATE,
 fertigsoll   DATE,
 astnr        INTEGER DEFAULT 1 CONSTRAINT astat_notnull NOT NULL,
 CONSTRAINT   a_knr_ref_kunde FOREIGN KEY (knr) REFERENCES kunde,
 CONSTRAINT   konr_ref_kontakt FOREIGN KEY (konr) REFERENCES kontakt,
 CONSTRAINT   astnr_ref_aufstat FOREIGN KEY (astnr) REFERENCES aufstat(astnr)
)
```

ANR ist der Primärschlüssel der Tabelle AUFTRAG. Der Fremdschlüssel KNR bezieht sich auf den Primärschlüssel KUNDE.KNR der Tabelle KUNDE, der Fremdschlüssel KONR bezieht sich auf den Primärschlüssel KONTAKT.KONR der Tabelle KONTAKT, und der Fremdschlüssel ASTNR bezieht sich auf den Primärschlüssel ASTNR der Tabelle AUFSTAT.
Die DEFAULT-Klausel bei der Spaltendefinition für ADATUM legt mit CURRENT_DATE als voreingestellten Wert das aktuelle Datum fest. Die DEFAULT-Klausel für ASTNR legt den Wert 1 als voreingestellten Wert fest. Die Tabelle AUFTRAG enthält folgende Daten:

anr	knr	konr	adatum	atext	fertigist	fertigsoll	astnr
200	102	25	2009-04-15	Mitarbeiterschulung	2009-05-02	2009-05-02	5
210	106	40	2009-12-15	Kunden-Verwaltung	2010-04-12	2010-04-01	3
211	106	41	2009-12-29	Datenbank-Entwurf Kunden	2010-04-09	2010-04-01	4
250	105	35	2010-01-19	Serienbrief-Einweisung		2010-03-03	2
251	105	35	2010-01-19	Kunden-Verwaltung		2010-05-02	2
300	101	20	2010-02-16	Netzwerk-Test/Vergleich			1
305	105	35	2010-04-28	Mitarbeiterschulung		2010-05-02	2

Tabelle 7: Daten der Tabelle AUFTRAG

Tabelle LEISTUNG

Die Tabelle LEISTUNG enthält die einzelnen zu einem Auftrag erbrachten Leistungen. Die Tabelle enthält eine eindeutige Leistungsnummer, die zugehörige Auftragsnummer, das Datum, an dem die Leistung erbracht wurde, die Bezeichnung der Leistung, die Leistungseinheit, Anzahl der Einheiten, Preis pro Einheit, zugehöriger Mehrwertsteuersatz und Rechnungsnummer.

Die Tabelle LEISTUNG wird mit der folgenden Tabellendefinition erzeugt:

```
CREATE TABLE     leistung
(lnr             INTEGER CONSTRAINT lnr_primary PRIMARY KEY,
 anr             INTEGER CONSTRAINT l_anr_notnull NOT NULL,
 ldatum          DATE,
 ltext           CHAR(25),
 leinheit        CHAR(10),
 lanz            INTEGER CONSTRAINT lanz_pos CHECK (lanz > 0),
 lsatz           NUMERIC (5,0),
 mwsatz          NUMERIC(2,2),
 rnr             NUMERIC(4,0),
 CONSTRAINT      anr_ref_auftrag FOREIGN KEY(anr) REFERENCES auftrag
)
```

LNR ist der Primärschlüssel der Tabelle LEISTUNG. Der Fremdschlüssel ANR bezieht sich auf den Primärschlüssel AUFTRAG.ANR der Tabelle AUFTRAG.

Die Tabelle LEISTUNG enthält folgende Daten:

lnr	anr	ldatum	ltext	leinheit	lanz	lsatz	mwsatz	rnr
1	200	2009-04-20	Schulungsmaterial	Seiten	45	75	0.19	3
2	200	2009-04-22	Schulung	Tag	1	1500	0.19	3
3	200	2009-04-23	Schulung	Tag	1	1500	0.19	3
4	211	2010-01-21	Systemanalyse	Tag	8	1200	0.00	10
5	211	2010-01-28	Datenbankentwurf	Tag	10	1200	0.00	10
6	211	2010-02-16	Kopien/Folien	Seiten	30	50	0.19	10
7	211	2010-03-24	Handbuch	Festpreis	1	200	0.07	10
10	250	2010-02-23	Reisekosten	Festpreis	2	125	0.00	
11	250	2010-02-23	Schulung	Tag	1	1200	0.19	

Tabelle 8: Daten der Tabelle LEISTUNG

3.3.3.2 Schema TEILE

Das Schema TEILE dient zur Teileverwaltung eines Fahrradversandes.
Das Schema TEILE enthält die Tabellen ARTIKEL, KATART, VERWENDUNG, LAGER, FARBTAB und TABTAB.

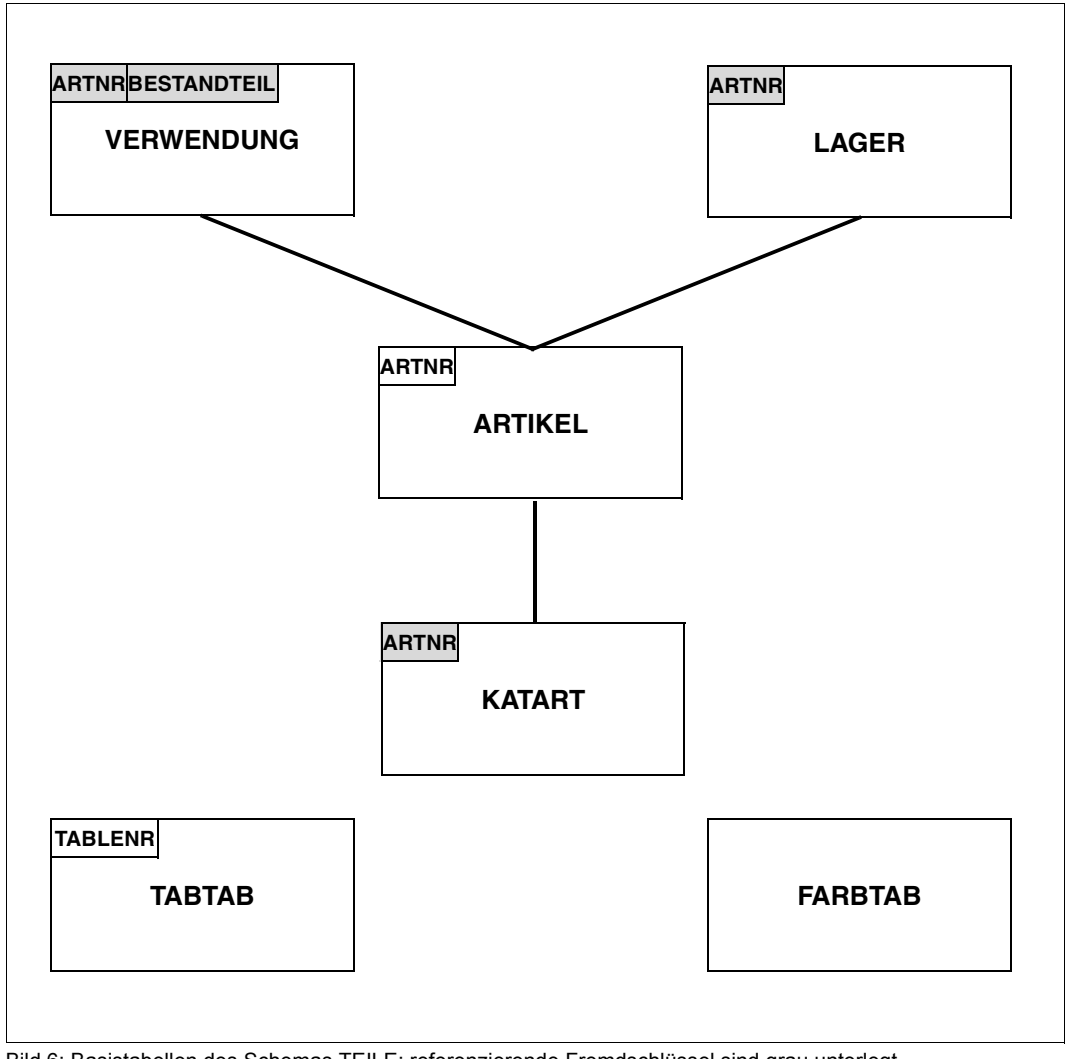


Bild 6: Basistabellen des Schemas TEILE; referenzierende Fremdschlüssel sind grau unterlegt

Tabelle ARTIKEL

Die Tabelle ARTIKEL enthält Informationen über die geführten Artikel. Sie besteht aus einer eindeutigen Artikelnummer, der Artikelbezeichnung, der Farbe des Artikel, dem Preis, dem aktuellen Bestand des Artikels und dem Mindestbestand dieses Artikels.

Die Tabelle ARTIKEL ist wie folgt definiert:

```
CREATE TABLE   artikel
(artnr         INTEGER CONSTRAINT artnr_primary PRIMARY KEY,
 artbez        CHARACTER(20) CONSTRAINT artbez_notnull NOT NULL,
 farbe         CHARACTER(15),
 preis         NUMERIC(8,2) CONSTRAINT preis_notnull NOT NULL,
 bestand       INTEGER CONSTRAINT a_bestand_notnull NOT NULL,
 minbestand    INTEGER
)
```

ARTNR ist der Primärschlüssel der Tabelle ARTIKEL.

Die Tabelle ARTIKEL enthält folgende Daten:

artnr	artbez	farbe	preis	bestand	minbestand
1	Fahrrad	schwarz	700.50	2	1
2	Fahrrad	feuerrot	230.00	1	1
10	Rahmen	schwarz	150.00	10	5
11	Rahmen	alpinweiss	150.00	10	5
120	Vorderrad	metallic	40.00	3	5
130	Hinterrad	metallic	40.00	12	5
200	Lenkstange	metallic	60.00	1	5
210	V-Nabe	metallic	5.00	15	1
220	H-Nabe	metallic	5.00	14	10
230	Felge	schwarz	10.00	9	10
240	Speiche	schwarz	1.00	211	240
500	Schraube M5	schwarz	1.10	300	240
501	Mutter M5	schwarz	0.75	295	240

Tabelle 9: Daten der Tabelle ARTIKEL

Tabelle KATART

Die Tabelle KATART enthält zwei REF-Spalten. Die REF-Werte in diesen Spalten verweisen auf BLOB-Objekte in den BLOB-Tabellen BILDER und BESCHREIBUNG im Schema ZUSAETZE. Die Tabelle KATART ist wie folgt definiert:

```
CREATE TABLE   katart
(artnr         INTEGER CONSTRAINT k_artnr_notnull NOT NULL,
 abb           FOR REF(bilder),
 beschr        FOR REF(beschreibung),
)
```

Die Tabelle KATART enthält folgende Daten:

artnr	abb	beschr
2	ZUSAETZE/BILDER? UID=942acb5a471511db8700a9de6f05052d &OID=1	ZUSAETZE/BESCHREIBUNG? UID=9c3d3d64471511db8600bb19c953d9f8 &OID=1
120	ZUSAETZE/BILDER? UID=a9f7656a471511db8f019d75c8d27c82 &OID=2	ZUSAETZE/BESCHREIBUNG? UID=b0e1d5cc471511db8300ae9379299332 &OID=2
500	ZUSAETZE/BILDER? UID=bdfe5e10471511db8600e510dc40a634 &OID=3	ZUSAETZE/BESCHREIBUNG? UID=c279d1ea471511db8a01e1958d1b9863 &OID=3
501	ZUSAETZE/BILDER? UID=cf7c2fbe471511db8e018591fbfe609b &OID=4	ZUSAETZE/BESCHREIBUNG? UID=d49b3a08471511db8c0096edddbf555c &OID=4

Tabelle 10: Daten der Tabelle KATART

Tabelle VERWENDUNG

Die Tabelle VERWENDUNG gibt an, aus welchen einzelnen Bestandteilen in welcher Anzahl ein Artikel zusammengesetzt ist. Einige Artikel werden einzeln und als Bestandteil eines anderen Artikels verwendet. Die Tabelle Verwendung enthält die Artikelnummer eines Artikels in der Tabelle ARTIKEL, die Artikelnummer des Bestandteils in der Tabelle ARTIKEL und wie oft dieser Bestandteil in dem Artikel vorhanden ist (ANZAHL).

Die Tabelle VERWENDUNG ist wie folgt definiert:

```
CREATE TABLE     verwendung
(artnr           INTEGER CONSTRAINT v_artnr_notnull NOT NULL,
 bestandteil     INTEGER CONSTRAINT bestandteil_notnull NOT NULL,
 anzahl          INTEGER CONSTRAINT anzahl_notnull NOT NULL,
 CONSTRAINT      v_artnr_ref_artikel FOREIGN KEY (artnr) REFERENCES artikel,
 CONSTRAINT      bestandteil_ref_artikel FOREIGN KEY (bestandteil)
                 REFERENCES artikel
)
```

Die Fremdschlüssel ARTNR und BESTANDTEIL beziehen sich auf den Primärschlüssel ARTIKEL.ARTNR der Tabelle ARTIKEL.

Die Tabelle VERWENDUNG enthält folgende Daten:

artnr	bestandteil	anzahl
1	10	1
1	120	1
1	130	1
1	200	1
120	210	1
120	230	1
120	240	15
120	500	5
120	501	5
200	500	10
200	501	10

Tabelle 11: Daten der Tabelle VERWENDUNG

Tabelle LAGER

Die Tabelle LAGER enthält Informationen über den Bestand der Artikel in einzelnen Lagern. Sie besteht aus der Artikelnummer in der Tabelle ARTIKEL und dem Bestand des Artikels an einem bestimmten Lagerort.

Die Tabelle LAGER ist wie folgt definiert:

```
CREATE TABLE     lager
(artnr           INTEGER CONSTRAINT l_artnr_notnull NOT NULL,
 bestand         INTEGER CONSTRAINT l_bestand_notnull NOT NULL,
 ort             CHAR(25),
 CONSTRAINT      l_artnr_ref_artikel FOREIGN KEY (artnr) REFERENCES artikel
)
```

Der Fremdschlüssel ARTNR bezieht sich auf den Primärschlüssel ARTIKEL.ARTNR der Tabelle ARTIKEL. Die Tabelle LAGER enthält folgende Daten:

artnr	bestand	ort
1	2	Hauptlager
2	1	Hauptlager
10	10	Hauptlager
11	10	Hauptlager
120	3	Hauptlager
130	3	Hauptlager
130	9	Teilelager
200	1	Hauptlager
210	15	Hauptlager
220	8	Hauptlager
220	6	Teilelager
230	6	Hauptlager
230	3	Teilelager
240	11	Hauptlager
240	200	Teilelager
500	120	Hauptlager
500	180	Teilelager
501	248	Hauptlager
501	47	Teilelager

Tabelle 12: Daten der Tabelle LAGER

Tabelle FARBTAB

Die Tabelle FARBTAB enthält Informationen darüber, wie einzelne Farben aus unterschiedlichen Anteilen an rot, gelb und blau erzeugt werden können. Sie besteht aus dem Namen der Farbe und den Farbanteilen an rot, gelb und blau.

Die Tabelle FARBTAB ist wie folgt definiert:

```
CREATE TABLE      farbtab
(farbname         CHARACTER(15),
 rgb              (3)NUMERIC(2,2)
)
```

Die Tabelle FARBTAB enthält folgende Daten:

farbname	**rgb**		
feuerrot	0.98	0	0
orange	0.9	0.3	0
himmelblau	0	0	0.99
aquamarinblau	0	0.1	0.99
alpinweiss	0.99	0.99	0.99
schwarz	0	0	0
metallic	0	0.2	0.3

Tabelle 13: Daten der Tabelle FARBTAB

Tabelle TABTAB

Die Tabelle TABTAB enthält Informationen über die Tabellen, die im Schema TEILE zusammengefasst werden. Sie besteht aus einer eindeutigen Tabellennummer, dem Tabellennamen und der Funktion der Tabelle.

Die Tabelle TABTAB ist wie folgt definiert:

```
CREATE TABLE       tabtab
(tablenr           INTEGER CONSTRAINT tablenr_primkey PRIMARY KEY,
 tablename         CHARACTER(20) CONSTRAINT tablename_notnull NOT NULL,
 bemerkung         CHARACTER(50)
)
```

TABLENR ist der Primärschlüssel der Tabelle TABTAB. Die Tabelle TABTAB enthält folgende Daten:

tablenr	**tablename**	**bemerkung**
1	artikel	Teiledaten
2	verwendung	Beziehungsdaten Fahrrad
3	lager	Lagerorte der Teile
4	farbtab	Erlaubte Farben
5	tabtab	Verwendete Tabellen
6	katart	Basisdaten für Katalog

Tabelle 14: Daten der Tabelle TABTAB

3.3.3.3 Schema ZUSAETZE

Das Schema ZUSAETZE enthält die beiden BLOB-Tabellen BILDER und BESCHREIBUNGEN. In diesen Tabellen werden BLOB-Objekte zweier unterschiedlicher Klassen gespeichert. Die Tabelleninhalte sind aufgrund der Struktur der BLOB-Tabellen nicht darstellbar.

BLOB-Tabelle BILDER

Die BLOB-Tabelle BILDER enthält BLOB-Objekte der Klasse Abbildungen. Die BLOB-Tabelle enthält die Abbildungen, auf die die REF-Spalte ABB der Tabelle KATART verweist. Die BLOB-Tabelle ist wie folgt definiert:

```
CREATE TABLE       bilder OF BLOB
(MIME              ('image/gif'),
 USAGE             ('Abbildungen für teile.katart.abb'),
                   'Fotograf: Hans Sesamer'
)
```

In der mitgelieferten Beispieldatenbank sind in der BLOB-Tabelle BILDER Abbildungen des Formats GIF abgespeichert, die mit gängigen Grafikprogrammen bearbeitet werden können. Beispielsweise ist folgende Abbildung des Fahrrads beigefügt:

BLOB-Tabelle BESCHREIBUNG

Die Tabelle BESCHREIBUNG enthält BLOB-Objekte der Klasse Word-Dokumente. Die BLOB-Tabelle enthält die Dokumente, auf die die REF-Spalte BESCHR der Tabelle KATART verweist. Die BLOB-Tabelle ist wie folgt definiert:

```
CREATE TABLE       beschreibung OF BLOB
(MIME              ('application/msword'),
 USAGE             ('worddokumente für teile.katart.beschr'),
                    '<AUTHOR>Herta Sesamer</AUTHOR>'
)
```

In der mitgelieferten Beispieldatenbank sind in der BLOB-Tabelle BESCHREIBUNG Word-Dokumente abgespeichert, die mit MS Word bearbeitet werden können. Für die Beschreibung des Fahrrads ist zum Beispiel folgendes Dokument beigefügt:

4 SQL-Funktionsumfang von SESAM/SQL

Dieses Kapitel stellt die wichtigsten SQL-Begriffe zusammen, die in den Handbüchern zu SESAM/SQL verwendet werden. Die vorgestellten Begriffe beziehen sich auf SQL-Objekte, die mit SQL-Anweisungen erzeugt und angesprochen werden können, oder sind weitere wichtige Begriffe, die im Zusammenhang mit SQL von Bedeutung sind. Allgemeine Kenntnisse über relationale Datenbanksysteme und SQL-Kenntnisse werden vorausgesetzt.

Die SQL-Funktionen werden nur insoweit dargestellt, als es für das Verständnis der restlichen Kapitel dieses Handbuchs notwendig ist. Die SQL-Anweisungen und -Sprachmittel werden nicht detailliert beschrieben. Für eine vollständige und systematische Darstellung der SQL-Anweisungen von SESAM/SQL wird auf die Handbücher „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“ und „SQL-Sprachbeschreibung Teil 2: Utilities“ verwiesen.

- Der Abschnitt „SQL-Objekte einer SESAM/SQL-Datenbank“ auf Seite 79 beschreibt die SQL-Objekte, die mit SQL-Anweisungen angesprochen werden können sowie weitere wichtige SQL-Begriffe. Zu einem bestimmten SQL-Begriff werden jeweils kurz die SQL-Anweisungen dargestellt, die im Zusammenhang mit diesem Begriff wichtig sind.
- Im Abschnitt „SQL-Anweisungen“ auf Seite 106 werden die einzelnen Klassen von SQL-Anweisungen in Form von Übersichten dargestellt.
- Der Abschnitt „Grundlegende SQL-Sprachmittel“ auf Seite 111 beschreibt die Sprachmittel, die bei SESAM/SQL in SQL-Anweisungen verwendet werden. Sie werden in knapper Form vorwiegend anhand von Beispielen dargestellt. Im Einzelnen behandelt der Abschnitt
  - die Namen von SQL-Objekten
  - welche Datentypen SESAM/SQL kennt
  - wie Werte, Ausdrücke und Funktionen in SQL angegeben werden
  - wie mit einer Suchbedingung Sätze aus Tabellen ausgewählt werden können
  - wie mit einem Abfrage-Ausdruck Sätze und Spalten aus Tabellen ausgewählt werden können
  - den SELECT-Ausdruck als elementaren Abfrage-Ausdruck
  - den Join-Ausdruck, mit dem Tabellen verbunden werden können
  - welche Anweisungen zur Datenmanipulation SQL zur Verfügung stellt.

- Der Abschnitt „SQL-Transaktion“ auf Seite 134 beschreibt die im Zusammenhang mit einer Transaktion wichtigen SQL-Anweisungen und -Begriffe.
- Im Abschnitt „Programmeinbettung von SQL“ auf Seite 138 wird beschrieben, wie der SESAM/SQL-Anwender
  - von einem COBOL-Programm aus auf eine SESAM/SQL-Datenbank zugreifen kann
  - mit Hilfe eines Cursors effizient auf die Sätze einer Tabelle zugreifen kann
  - flexible Anwendungen mit Hilfe von dynamischen - d.h. erst zur Laufzeit übersetzten SQL-Anweisungen - realisieren kann.

Wichtige Konzepte für den Zugriffsschutz in SQL werden im Abschnitt „Zugriffsschutz in SQL durch Privilegien“ auf Seite 176 dargestellt.

4.1 SQL-Objekte einer SESAM/SQL-Datenbank

Dieser Abschnitt beschreibt die SQL-Objekte, die mit SQL-Anweisungen angesprochen werden können.

„SQL-Objekte“ ist der Oberbegriff für alle Elemente der Datenbank, die mit SQL-Anweisungen erzeugt und - mit Ausnahme der Privilegien (siehe Seite 176) - auch benannt werden können.

Im Einzelnen behandelt dieser Abschnitt folgende SQL-Objekte:

- SESAM/SQL-Datenbank/(Catalog)
- Space
- Storage Group
- Schema
- Tabelle
- Spalte
- Integritätsbedingungen
- Index
- Routinen (Prozeduren und User Defined Functions (UDF))
- BLOB-Konstrukte

Bild 7 auf Seite 80 veranschaulicht den Zusammenhang einiger grundlegender SQL-Objekte für die Beispieldatenbank AUFTRAGKUNDEN (siehe Seite 16):

Die Metadaten der **Datenbank** AUFTRAGKUNDEN enthalten das **Schema** AUFTRAGSVER. Teile der Metadaten sind dem Anwender über das **Schema** INFORMATION_SCHEMA zugänglich. Im Schema AUFTRAGSVER befindet sich die Definition der **Tabelle** KUNDE. Der Anwender kann festlegen, auf welchen **Anwender-Spaces** (BS2000-Dateien) die Tabellen mit den Anwenderdaten angelegt werden. Die Metadaten befinden sich auf dem **Catalog-Space**. Die Tabelle KUNDE wird auf dem TABLESPACE und der **Index** INDKINFO auf dem INDEXSPACE angelegt. Anwender-Spaces und Catalog-Space können wiederum **Storage Groups** zugeordnet werden, die die physikalischen Speichermedien zusammenfassen, auf denen die Space-Dateien angelegt werden. TABLESPACE und INDEXSPACE werden der Storage Group STOGROUP1 und der Catalog-Space der Storage Group STOGROUP2 zugeordnet.

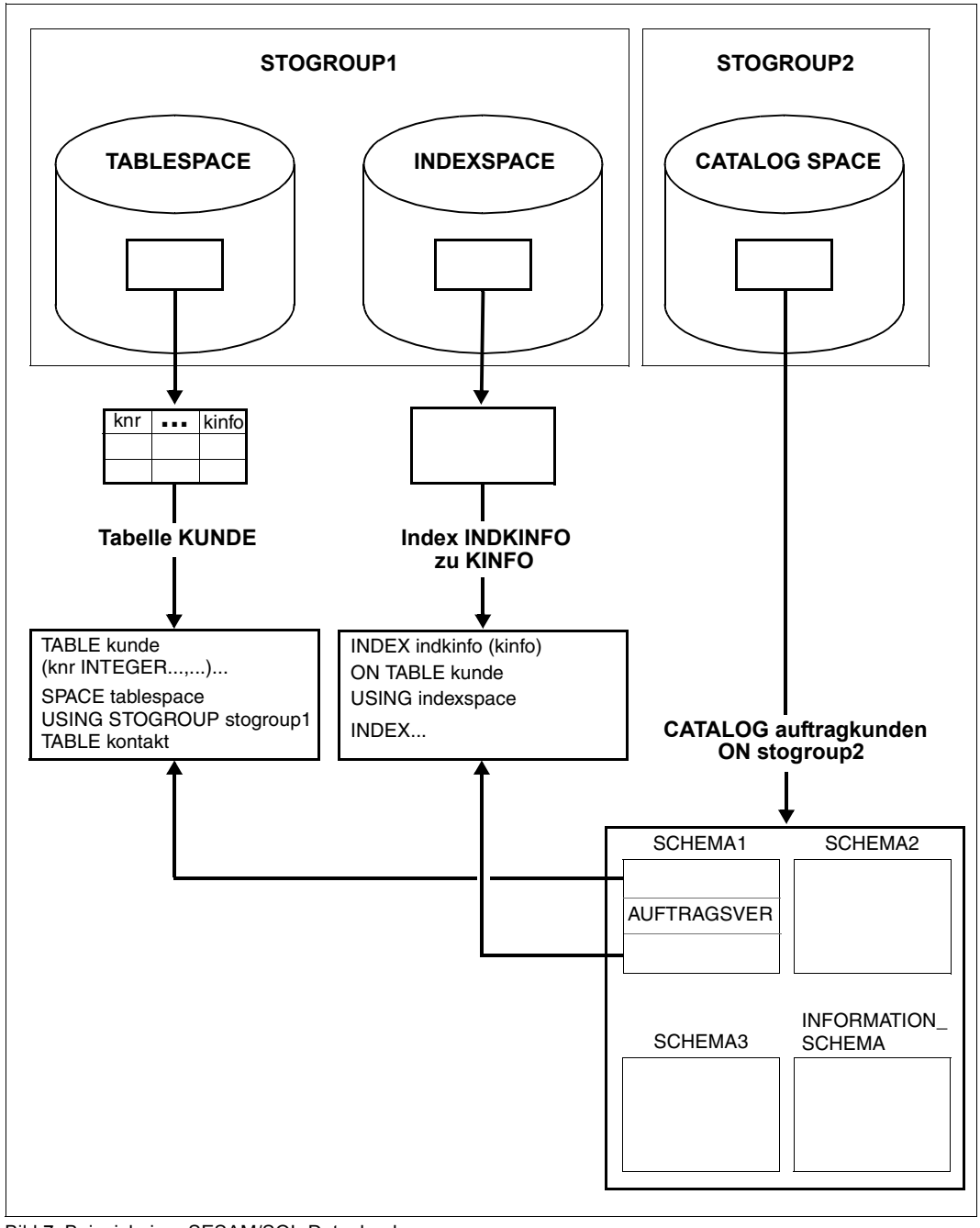


Bild 7: Beispiel einer SESAM/SQL-Datenbank

4.1.1 SESAM/SQL-Datenbank

Eine Datenbank (synonym: ein Katalog) lässt sich als Menge von zusammengehörigen Datenbeständen auffassen, die mit Hilfe eines Datenbanksystems verwaltet werden.

Bei SESAM/SQL besteht eine Datenbank aus den *Metadaten* im *Catalog-Space* und den *Anwenderdaten* in den zugehörigen *Anwender-Spaces*. Eine Datenbank wird durch den Datenbanknamen identifiziert.

Anwenderdaten

Eine relationale Datenbank kann aus Benutzersicht als eine Menge von Tabellen aufgefasst werden. Der SQL-Benutzer ist für den Aufbau und den Inhalt der Tabellen verantwortlich, die er mit SQL-Anweisungen definiert. Aus seiner Sicht handelt es sich bei diesen Tabellen um Anwenderdaten.

Metadaten

Neben diesen Anwenderdaten benötigt ein Datenbanksystem interne Daten, die die Struktur der Anwenderdaten beschreiben und die zum Verwalten der Anwenderdaten benötigt werden. Diese internen Daten werden auch als Metadaten bezeichnet. Die Metadaten einer Datenbank sind im Catalog-Space der Datenbank abgelegt. Teile der Metadaten sind dem Anwender über das Schema INFORMATION_SCHEMA zugänglich.

Space

Anwenderdaten und Metadaten befinden sich auf sogenannten Spaces. Spaces sind BS2000-Dateien. Sie spielen eine zentrale Rolle bei der Sicherung und Wiederherstellung von Datenbanken (siehe Seite 199).

Man unterscheidet Anwender-Spaces, in denen die Anwenderdaten, nämlich Tabellen und Indizes, gespeichert sind, vom Catalog-Space, der die Metadaten enthält. Aus physikalischer Sicht besteht eine SESAM/SQL-Datenbank aus den Anwenderdaten in den entsprechenden Anwender-Spaces und den zugehörigen Metadaten im Catalog-Space.

Ein Space kann auf Pubsets mit „großen Dateien“ bis zu 4 TByte groß werden.
Sonst kann er bis zu 64 GByte groß werden.

Datenbank (Katalog)

Eine Datenbank wird erzeugt, indem mit der Utility-Anweisung CREATE CATALOG (siehe Seite 341) der Catalog-Space der Datenbank erzeugt wird. Dabei wird der Name der Datenbank festgelegt. Die Namen aller Objekte, die zu der Datenbank mit diesem Catalog-Space gehören, werden mit dem Datenbanknamen qualifiziert.

Um die Portierbarkeit von Anwenderprogrammen zu erhöhen (z.B. Wechsel von Test- und Produktivumgebung), unterscheidet SESAM/SQL den logischen und den physikalischen Datenbanknamen. Der physikalische Datenbankname ist der Name einer existierenden SESAM/SQL-Datenbank.

Der logische Datenbankname ist der Name, mit dem ein Anwenderprogramm eine SESAM/SQL-Datenbank anspricht. Existiert zu dem logischen Datenbanknamen keine SESAM/SQL-Datenbank, muss die Zuordnung zu einer existierenden Datenbank über den physikalischen Datenbanknamen im SQL-Datenbankverzeichnis erfolgen. Bei Qualifizierung von SQL-Objekten mit dem Datenbanknamen wird stets der logische Datenbankname verwendet.

Anwenderspace

Ein Anwender-Space wird durch die SQL-Anweisung CREATE SPACE erzeugt, siehe Seite 350.

In den Anweisungen CREATE TABLE und CREATE INDEX kann dann der Name des Anwender-Space angegeben werden, um die Tabelle bzw. den Index auf diesem Space anlegen zu lassen. Eine bestimmte Basistabelle oder ein bestimmter Index muss vollständig in einem Space enthalten sein.

Mit ALTER SPACE können Eigenschaften des Catalog-Space oder eines Anwender-Space und der Name der zugehörigen Storage Group verändert werden.

DROP SPACE löscht einen Anwender-Space.

4.1.2 Storage Group

Beim Aufbau einer SESAM/SQL-Datenbank muss sich der Anwender nicht darum kümmern, auf welchen Datenträgern Anwenderdaten und Metadaten einer Datenbank angelegt werden. Es kann jedoch sinnvoll sein, dass der Anwender die Verteilung der Spaces auf verschiedene Datenträger selbst beeinflusst, indem er z.B. Daten, die häufig gemeinsam angesprochen werden, benachbart ablegt, oder Daten, auf die häufig zugegriffen wird, auf Platten mit kurzen Zugriffszeiten verlagert. Diese Möglichkeit hat der SESAM/SQL-Anwender, indem er festlegen kann, auf welchen Platten die Spaces einer Datenbank abgelegt werden.

Dazu kann der Anwender mit der SQL-Anweisung CREATE STOGROUP eine sogenannte Storage Group erzeugen.

Eine Storage Group fasst Speichermedien einer BS2000-Katalogkennung unter einem Namen zusammen. Alle Speichermedien einer Storage Group müssen denselben Gerätetyp haben. Die Spaces einer Datenbank können auf verschiedene Storage Groups verteilt werden.

Wurde mit einer CREATE STOGROUP-Anweisung eine Storage Group erzeugt, dann können mit nachfolgenden CREATE SPACE-Anweisungen dieser Storage Group Anwender-Spaces zugeordnet werden. In den Anweisungen CREATE TABLE und CREATE INDEX legt der Anwender dann fest, auf welchem Space eine bestimmte Tabelle bzw. ein bestimmter Index angelegt wird.

Die Storage Group des Catalog-Space einer Datenbank legt der Anwender in der Utility-Anweisung CREATE CATALOG fest. Aus dem Datenbank-Namen wird der Name des Catalog-Space erzeugt: :*catid*:*system-benutzerkennung*.*catalog*.CATALOG

Wird bei CREATE STOGROUP keine Katalogkennung festgelegt, dann wird die Katalogkennung des Standard-Pubset der DBH-Session verwendet.

Mit ALTER STOGROUP kann die Definition einer Storage Group geändert werden. So können neue Speichermedien hinzugefügt oder einzelne Speichermedien aus der Storage Group entfernt werden.

DROP STOGROUP löscht eine Storage Group, wenn sie nicht mehr von einem Space benutzt wird.

4.1.3 Schema

Eine Datenbank ist in sogenannte Schemata aufgeteilt. Man unterscheidet anwenderdefinierte Schemata und Informationsschemata.

Anwenderdefiniertes Schema

Jedes anwenderdefinierte Schema innerhalb einer Datenbank ist einem Benutzer zugeordnet, der Eigentümer des Schemas ist. Ein anwenderdefiniertes Schema enthält Metadaten, die den formalen Aufbau der Basistabellen, Views, Indizes, Integritätsbedingungen, Privilegien und Routinen beschreiben, die der Eigentümer des Schemas definiert.

Jedes anwenderdefinierte Schema hat einen Namen und einen Eigentümer, der durch einen sogenannten Berechtigungsschlüssel (siehe Seite 174) für dieses Schema ausgewiesen ist. Es wird durch die SQL-Anweisung CREATE SCHEMA erstellt und kann durch andere SQL-Anweisungen zur Schemadefinition und -verwaltung modifiziert werden.

Die Anweisungen CREATE TABLE, CREATE VIEW, CREATE PROCEDURE, GRANT und CREATE INDEX können als Teile der CREATE SCHEMA-Anweisung oder als eigenständige Anweisungen angegeben werden. Eine Voraussetzung für alle SQL-Anweisungen zur Schemadefinition und -verwaltung ist, dass die DBH-Startanweisung ADD-SQL-DATABASE-CATALOG-LIST (siehe Handbuch „Datenbankbetrieb“) mit dem Parameter ACCESS=*PARAMETERS (CAT-ADMINISTRATION=*YES) gegeben wurde.

Ein Schema wird mit der SQL-Anweisung DROP SCHEMA gelöscht.

Informationsschemata

Neben anwenderdefinierten Schemata besitzt jede Datenbank zwei sogenannte Informationsschemata mit den Namen INFORMATION_SCHEMA und SYS_INFO_SCHEMA.

Das INFORMATION_SCHEMA besteht aus Tabellen, die einen Teil der Metadaten einer Datenbank enthalten. Jeder Anwender kann diese Informationen in einem ESQL-Programm oder mit Hilfe des Utility-Monitors abfragen.

Das SYS_INFO_SCHEMA enthält systemspezifische Daten und ist nur dem universellen Benutzer (siehe Seite 173) zugänglich. Die Tabellen der Informationsschemata sind im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“ beschrieben.

4.1.4 Tabelle

Die Daten einer SESAM/SQL-Datenbank sind in Tabellen organisiert. Eine Tabelle ist eine zweidimensionale Anordnung von Daten, die aus Zeilen und Spalten besteht.

Die Zeilen einer Tabelle werden auch als Sätze bezeichnet. Die Reihenfolge der Sätze ist in einer Tabelle nicht geordnet. Auch die Anzahl der Sätze einer Tabelle ist nicht festgelegt.

Jeder Satz einer Tabelle hat dieselbe Anzahl von Spalten. Der Name und der Datentyp jeder Spalte sowie die Reihenfolge der Spalten werden durch die Tabellendefinition festgelegt. Alle Werte einer bestimmten Spalte besitzen denselben Datentyp.
Während eine Tabelle nach dem relationalen Modell keine doppelten Sätze enthalten darf, ist dies in SQL grundsätzlich möglich. SQL kennt jedoch Sprachmittel, mit denen das Auftreten doppelter Sätze unterdrückt werden kann.

Man unterscheidet folgende Arten von Tabellen:

- Basistabellen
- partitionierte Tabellen (Basistabellen, die auf mehrere Anwender-Spaces verteilt sind)
- Views
- Ergebnistabellen
- abstrakte Tabellen
- „read-only“ Tabellen

4.1.4.1 Basistabelle

Basistabellen sind Tabellen, die die Anwenderdaten einer Datenbank enthalten. Eine Basistabelle wird mit einer CREATE TABLE-Anweisung definiert und permanent in der Datenbank gespeichert, bis sie mit einer DROP TABLE-Anweisung gelöscht wird.

Mit der ALTER TABLE-Anweisung können in einer bestehenden Basistabelle Tabellenbedingungen (siehe Abschnitt „Integritätsbedingung“ auf Seite 93) hinzugefügt oder gelöscht und Spalten und Indizes hinzugefügt, Spalten geändert oder gelöscht werden.

SESAM/SQL unterscheidet zwischen den folgenden Arten von Basistabellen:

- Tabellen, die nur mit SQL bearbeitet werden können (SQL-Tabellen)
- Tabellen, die ausschließlich BLOB-Objekte enthalten und nur mit SQL bearbeitet werden können (BLOB-Tabellen)
- Tabellen, die nur mit CALL-DML bearbeitet werden können (Nur-CALL-DML-Tabellen)
- Tabellen, die mit CALL-DML und eingeschränkter SQL bearbeitet werden können (CALL-DML/SQL-Tabellen)

SQL-Tabellen, BLOB-Tabellen und CALL-DML/SQL-Tabellen können auch als partitionierte Tabellen erzeugt werden, siehe Abschnitt „Partitionierte Tabelle“ auf Seite 87.

Nur-CALL-DML-Tabellen und CALL-DML/SQL-Tabellen werden unter dem Namen CALL-DML-Tabellen zusammengefasst.

Die Tabellenart ist im Zusammenhang mit der Utility-Anweisung MIGRATE von Bedeutung. Mit der MIGRATE-Anweisung werden Datenbanken, die mit SESAM/SQL V1.1 bzw. einer früheren Version erstellt worden sind, in Basistabellen einer SESAM/SQL-Datenbank der aktuellen Version überführt.

Bei CALL-DML-Tabellen sind einige Besonderheiten zu beachten:
Für CALL-DML-Tabellen sind nur die Datentypen CHARACTER, NUMERIC, DECIMAL, INTEGER und SMALLINT erlaubt. Für eine Spalte darf mit DEFAULT kein voreingestellter Wert definiert werden.

Als voreingestellter Wert darf nur der nicht-signifikante Wert in der CALL-DML-Klausel angegeben werden. CALL-DML-Tabellen müssen eine Primärschlüsselbedingung (siehe Seite 94) für einen einfachen oder zusammengesetzten Primärschlüssel besitzen. Der Name der Primärschlüsselbedingung wird als Name des zusammengesetzten Primärschlüssels verwendet. Außer dieser Primärschlüsselbedingung sind keine Integritätsbedingungen erlaubt.

Speicherstruktur von Basistabellen

Wenn eine Tabelle mit CREATE TABLE angelegt wird, dann wird Speicherplatz, der sogenannte zusammenhängende Bereich der Tabelle, reserviert. Die einzufügenden Sätze werden in diesem Bereich gespeichert. Wenn ein Satz eingefügt oder erweitert werden soll und der Platz in diesem Bereich nicht mehr ausreicht, dann wird eine sogenannte Auslagerung, ein freier Block, angelegt. Dieser Block gehört logisch zur Tabelle, liegt aber physikalisch nicht mehr im zusammenhängenden Bereich der Tabelle.
Bei der nächsten Reorganisation des Anwender-Space mit der Anweisung REORG SPACE werden alle existierenden Tabellen und Indizes auf dem Anwender-Space neu aufgebaut. Damit verschwinden auch die Auslagerungen.

Basistabelle in eine partitionierte Tabelle umwandeln

Eine nicht-partitionierte Basistabelle mit Primärschlüssel kann mit der Utility-Anweisung ALTER PARTITIONING FOR TABLE in eine partitionierte Tabelle umgewandelt werden, siehe Abschnitt „Partitionierung einer Basistabelle ändern“ auf Seite 379 und das Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“.

4.1.4.2 Partitionierte Tabelle

Eine partitionierte Tabelle ist eine Basistabelle, deren Daten in mehreren Anwender-Spaces gespeichert sind. Die auf einem Space liegenden Daten der Tabelle werden als Partition der Tabelle bezeichnet. Alle Partitionen einer Tabelle müssen auf unterschiedlichen Spaces liegen. Es sind 2 bis 16 Partitionen pro Tabelle möglich. In SESAM/SQL werden die Daten satzweise auf die Partitionen aufgeteilt, das Zuordnungskriterium ist der Primärschlüsselwert eines Satzes.

Partitionierte Tabellen haben gegenüber nicht-partitionierten Tabellen folgende Vorteile:

- Die Anwenderdaten können nach bestimmten Kriterien, z.B. monatsweise, übersichtlich strukturiert und in unterschiedlichen Partitionen bzw. Anwender-Spaces abgelegt werden. Eine Partition umfasst dann die Anwenderdaten, auf die aktuell zugegriffen wird.
- Bei geeigneter Strukturierung bzw. Partitionierung wird die Größe der Anwender-Spaces, auf die aktuell zugegriffen wird, verkleinert.

  Ein Zugriff auf die Anwender-Spaces, die aktuell benötigt werden, ist auch dann möglich, wenn weitere Partitionen bzw. Anwender-Spaces der Tabelle nicht verfügbar sind, siehe Abschnitt „Teilverfügbarkeit“ auf Seite 89.
- Die Sicherungs- und Reparaturzeiten können verkürzt werden, da die einzelnen Anwender-Spaces kleiner werden.
- Die Tabelle kann größer als 64 GB sein.

Die folgende Übersicht stellt die wichtigsten Eigenschaften einer nicht-partitionierten und einer partitionierten Tabelle gegenüber:

Eigenschaft	**nicht-partitionierte Tabelle**	**partitionierte Tabelle**
Anzahl Anwender-Spaces	1	2 bis 16 (alle Spaces müssen disjunkt und vorher angelegt sein)
Primärschlüssel	optional	obligatorisch (ein- oder mehrspaltig möglich)
Primärschlüsselwerte eines existierenden Satzes mit UPDATE oder MERGE ändern	erlaubt	nicht erlaubt (Satz muss gelöscht und mit geändertem Wert neu eingefügt werden)
Maximale Anzahl Sätze	ca. 4,3 Mrd.	ca. 268 Mio pro Partition, bei 16 Partitionen max. 4,3 Mrd. Sätze

Tabelle 15: Eigenschaften nicht-partitionierter und partitionierter Tabellen (Teil 1 von 2)

Eigenschaft	nicht-partitionierte Tabelle	partitionierte Tabelle
Kleinste Sicherungs- und Reparatureinheit	gesamte Tabelle	eine Partition (bzw. der dazu gehörende Anwender-Space)
Metadaten	INFORMATION_SCHEMA: – BASE_TABLES	INFORMATION_SCHEMA: – BASE_TABLES [1] – PARTITIONS
	SYS_INFO_SCHEMA: – SYS_TABLES	SYS_INFO_SCHEMA: – SYS_TABLES [1] – SYS_PARTITIONS
Nur-CALL-DML-Tabellen	möglich	nicht möglich
Passwortschutz mit SEPA	möglich	nicht möglich

Tabelle 15: Eigenschaften nicht-partitionierter und partitionierter Tabellen (Teil 2 von 2)

[1] Beim Spacenamen wird „_PARTITIONS_“ eingetragen

Weitere Eigenschaften von partitionierten Tabellen

- Die Partitionierung einer partitionierten Tabelle kann mit der Utility-Anweisung ALTER PARTITIONING FOR TABLE geändert oder aufgehoben werden, siehe Abschnitt „Partitionierung einer Basistabelle ändern“ auf Seite 379 und das Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“.

  Partitionsgrenzen können auch dadurch geändert werden, dass Sie die Tabelle in eine Exportdatei exportieren, die bestehende Tabelle mit DROP TABLE löschen und dann die Exportdatei als partitionierte Tabelle mit gleichem oder unterschiedlichem Namen und anderen Partitionsgrenzen importieren.

- Die Definition von Spalten, Integritätsbedingungen, Indizes und Defaultwerten bezieht sich immer auf alle Partitionen.
- Implizit angelegte Indizes oder explizit ohne USING SPACE Klausel angelegte Indizes werden immer auf dem Anwender-Space der ersten Partition gespeichert.
- Fehlerdateien, die von SESAM/SQL angelegt werden, werden bei CHECK FORMAL für jeden betroffenen Space angelegt. Bei LOAD und UNLOAD wird eine Fehlerdatei für alle Partitionen angelegt.
- Auf einem Anwender-Space, auf dem eine Partition gespeichert ist, dürfen andere Basistabellen, Indizes oder eine Partition einer anderen Tabelle gespeichert werden. Dieses Vorgehen wird aber nicht empfohlen, da sonst der Vorteil „kleine Sicherungs- und Reparatureinheiten“ geringer wird.

Teilverfügbarkeit

Für viele Arten von Zugriffen auf eine geeignet partitionierte Tabelle müssen nur die Anwender-Spaces, die aktuell benötigt werden, verfügbar sein. Weitere Partitionen bzw. Anwender-Spaces der Tabelle müssen nicht verfügbar sein. Dies wird als Teilverfügbarkeit von Partitionen bezeichnet.

Zusätzlich wird zwischen physikalischer und logischer Verfügbarkeit einer Partition unterschieden.

Physikalische und logische Verfügbarkeit

Physikalisch verfügbar heißt, dass auf den zur Partition gehörenden Space physikalisch zugegriffen werden kann und der Space im Zustand „space o.k." ist.

Logisch verfügbar heißt, dass der zur Partition gehörende Space SESAM/SQL-intern als physikalisch verfügbar vermerkt ist. Dieser Vermerk dient dazu, die Zugriffe zu beschleunigen, da nur der Vermerk und nicht der Space selber geprüft werden. Der Vermerk wird angelegt bzw. aktualisiert, wenn zum ersten Mal innerhalb einer DBH-Session per DML-Anweisung auf eine partitionierte Tabelle zugegriffen wird. Entsprechendes gilt für den ersten Zugriff nach dem Eintragen einer Datenbank in das SQL-Tabellenverzeichnis oder einem DBH-Start oder DBH-Restart.

Die physikalische und logische Verfügbarkeit spielt insbesondere bei DML-, DDL- und Utility-Anweisungen eine Rolle:

- DML-Anweisungen setzen voraus, dass die betroffenen Partitionen logisch verfügbar sind. Entscheidend ist die durchsuchte Datenmenge, nicht die Treffermenge. D.h. es müssen alle Partitionen, die von einem Suchvorgang betroffen sind, logisch verfügbar sein. Beim Einfügen eines Satzes ist es z.B. möglich, dass auch eine benachbarte Partition durchsucht werden muss.
- DDL-Anweisungen setzen voraus, dass die betroffenen Partitionen physikalisch verfügbar sind. Bei den Anweisungen CREATE INDEX, DROP INDEX, ALTER TABLE und DROP TABLE müssen alle Partitionen physikalisch verfügbar sein. Bei der Anweisung CREATE TABLE müssen alle Spaces, auf denen die partitionierte Tabelle angelegt werden soll, physikalisch verfügbar sein.
- Utility-Anweisungen setzen in den meisten Fällen voraus, dass die betroffenen Partitionen physikalisch verfügbar sind. Eine Ausnahme bilden die Anweisungen LOAD ONLINE, UNLOAD ONLINE ... WHERE, EXPORT ... WHERE und CHECK CONSTRAINTS.
  Einzelheiten zur Verfügbarkeit von Partitionen bei Utility-Anweisungen finden Sie im Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities".

Logische Verfügbarkeit ändern

Die logische Verfügbarkeit einer partitionierten Tabelle bleibt während einer DBH-Session unverändert, auch dann, wenn sich die physikalische Verfügbarkeit von Partitionen ändert. Nur nach RECOVER SPACE wird die logische und physikalische Verfügbarkeit aller angegebenen Anwender-Spaces neu bestimmt.

Mit den folgenden Administrationsanweisungen können Sie sich informieren und die logische Verfügbakeit ändern:

- Mit SHOW-PARTITIONS können Sie sich über die logische Verfügbarkeit der einzelnen Partitionen einer partitionierten Tabelle informieren.
- Mit CLOSE-SPACE setzen Sie alle auf diesem Space liegenden Partitionen auf „logisch nicht verfügbar“.
- Mit REUSE-PARTITIONS aktualisieren Sie die logische Verfügbarkeit der Partitionen einer partitionierten Tabelle. Damit werden Partitionen, die bislang zwar physikalisch aber nicht logisch verfügbar sind, auf „logisch verfügbar“ gesetzt.

4.1.4.3 View

Ein View ist eine Tabelle, die dem Benutzer eine definierte Sicht auf andere Tabellen der Datenbank ermöglicht. Anders als eine Basistabelle ist ein View nicht permanent in der Datenbank gespeichert. Sein Inhalt wird erst bei Bedarf bestimmt. Die Daten eines View sind nur in den zu Grunde liegenden Basistabellen vorhanden.

Somit enthält der View stets die Werte, die zum Zeitpunkt seiner Auswertung in der Datenbank enthalten sind.

Die Vorteile von Views sind Flexibilität bei der Datenbankabfrage, Speicherplatzersparnis und Möglichkeiten für einen abgestuften Datenschutz:

- Durch geeignet definierte Views lassen sich Daten zusammenstellen, die auf das Informationsbedürfnis der jeweiligen Benutzer zugeschnitten sind.
- Für Daten, die in verschiedenen Kombinationen benötigt werden, muss nicht jedesmal neu Speicherplatz angelegt werden.
- Nicht-privilegierten Benutzern kann der Zugriff auf Daten der Datenbank nur über geeignete Views für ausgewählte Daten gestattet werden (siehe Seite 191).

Ein View wird mit der CREATE VIEW-Anweisung erzeugt und mit DROP VIEW gelöscht. Wird der View unter seinem Namen abgefragt, dann stellt er sich für den Benutzer wie eine Basistabelle dar.

Mit Hilfe von änderbaren Views können Zeilen in zu Grunde liegenden Basistabellen eingefügt, geändert und gelöscht werden. Ein View ist änderbar, wenn in der CREATE VIEW-Anweisung ein Abfrage-Ausdruck angegeben ist und der zu Grunde liegende Abfrage-Ausdruck änderbar ist (siehe Seite 128).

4.1.4.4 Ergebnistabelle

Eine Ergebnistabelle ist eine Tabelle, die als Ergebnis der Auswertung eines Abfrage-Ausdrucks entsteht. Im Gegensatz zu Basistabellen und Views hat eine Ergebnistabelle keinen Namen. Die Werte der Ergebnistabelle werden zum Zeitpunkt der Auswertung des Abfrage-Ausdrucks aus den Werten der zu Grunde liegenden Tabellen ermittelt.

4.1.4.5 Abstrakte Tabelle

Eine abstrakte Tabelle (abstract table) ist eine Basistabelle, deren einzelne Zeilen nicht persistent gespeichert werden. SESAM/SQL berechnet diese Zeileneinträge aufgrund von Werten anderer Tabellen erst dann, wenn die Tabellen durch SESAM/SQL verwendet werden. Abstrakte Tabellen geben immer die momentan aktuellen Werte aus. In SESAM/SQL dienen abstrakte Tabellen als Grundlage für Views im Informationsschema.

4.1.4.6 „Read-only“ Tabellen

Tabellenfunktionen liefern „read-only“ Tabellen, deren Inhalt nicht von persistent gespeicherten SQL-Daten abhängt.

- Die Tabellenfunktion DEE() liefert eine Tabelle mit einer Zeile ohne Spalten
- Bei der Tabellenfunktion CSV() werden die Werte einer Tabelle aus einer BS2000-Datei gelesen

4.1.5 Spalte

Die Spaltendefinition legt bei der Erzeugung oder Änderung einer Basistabelle (CREATE TABLE, ALTER TABLE) den Namen und die Eigenschaften einer Spalte fest.

Jede Spalte besitzt einen Namen und einen Datentyp. Welche Datentypen in SESAM/SQL angegeben werden können, wird im Abschnitt „Datentypen“ auf Seite 114 beschrieben.

SESAM/SQL unterscheidet zwischen einfachen und multiplen Spalten.
Bei einer einfachen Spalte kann pro Satz genau ein Wert gespeichert werden.
Bei einer multiplen Spalte können pro Satz mehrere Werte desselben Datentyps gespeichert werden. Eine multiple Spalte besteht aus mehreren Spaltenelementen. In jedem Spaltenelement kann pro Satz genau ein Wert gespeichert werden.

Der Wert einer multiplen Spalte in einem Datensatz heißt Aggregat, der Wert eines einzelnen Spaltenelements in einem Datensatz heißt Ausprägung. Ein Aggregat setzt sich aus den Ausprägungen der einzelnen Spaltenelemente zusammen.
Von einer multiplen Spalte kann entweder ein Spaltenelement oder ein zusammenhängender Bereich von Spaltenelementen referenziert werden. Ein einzelnes Spaltenelement wird in der Form *spalte*[*n*] (oder *spalte*(*n*)), ein Bereich in der Form *spalte*[*n..m*] (oder *spalte*(*n..m*)) angegeben, wobei $m>n>0$ gilt.

Um BLOB-Objekte in „normale“ SQL-Basistabellen einzubinden, wird mit der FOR REF-Klausel eine REF-Spalte definiert. Diese kann dann Referenzen enthalten, die auf BLOB-Werte einer BLOB-Tabelle weisen. Ausführlich werden BLOB-Objekte, -Tabellen und REF-Werte im Abschnitt „BLOB-Konstrukte“ auf Seite 99 beschrieben.

In einer Spaltendefinition kann ein voreingestellter Wert für eine einfache Spalte festgelegt werden. Erhält beim Einfügen eines Satzes eine einfache Spalte keinen Wert, dann wird der Spalte der voreingestellte Wert zugewiesen. Als Voreinstellung kann ein Literal, eine Datumsfunktion, eines der Spezial-Literale CURRENT_USER oder SYSTEM_USER oder der NULL-Wert angegeben werden. Die Voreinstellung muss mit dem Datentyp der Spalte kompatibel sein und eine evtl. vorhandene Integritätsbedingung erfüllen. Die REF-Spalte erhält von SESAM/SQL als Voreinstellung den REF-Wert der Klasse. Dieser bezeichnet die gesamte Klasse der BLOB-Werte der in der FOR REF-Klausel angegebenen BLOB-Tabelle.

In der CONSTRAINT-Klausel kann eine Integritätsbedingung (siehe den Abschnitt „Integritätsbedingung“ auf Seite 93) für die Spalte definiert werden.

Bei CALL-DML-Tabellen wird in der CALL-DML-Klausel der nicht signifikante Wert und der Spaltenname angegeben.

4.1.6 Integritätsbedingung

Eine Integritätsbedingung ist eine Regel, die den Wertebereich für eine Spalte oder für mehrere Spalten einschränkt. Ähnlich wie nur Werte in die Datenbank aufgenommen werden, die mit dem für die entsprechende Spalte definierten Datentyp kompatibel sind, lässt SESAM/SQL nur Werte zu, die den definierten Integritätsbedingungen genügen. Eine Integritätsbedingung kann als eine für eine bestimmte Spalte oder mehrere Spalten formulierte Suchbedingung aufgefasst werden, deren Wahrheitswert nie falsch werden darf. Ein Satz kann nur in eine Tabelle aufgenommen oder aus einer Tabelle gelöscht werden und ein Spaltenwert kann nur geändert werden, wenn alle zugehörigen Integritätsbedingungen auch nach der Änderung noch erfüllt sind.

Eine Integritätsbedingung kann bei der Tabellendefinition mit CREATE TABLE definiert werden. Mit ALTER TABLE kann eine neue Integritätsbedingung für eine bestehende Basistabelle hinzugefügt oder eine Integritätsbedingung gelöscht werden.

Eine Integritätsbedingung kann als Tabellenbedingung oder als Spaltenbedingung angegeben werden. Eine Tabellenbedingung ist eine Integritätsbedingung, die für eine Spalte oder für eine Kombination von Spalten vereinbart wird. Bezieht sich die Integritätsbedingung auf nur eine Spalte, dann kann sie als Spaltenbedingung direkt bei der Definition der betreffenden Spalte angegeben werden.

Eine Integritätsbedingung hat einen Namen, der bei der Definition der Integritätsbedingung explizit vergeben werden kann oder implizit von SESAM/SQL vergeben wird. Falls eine Integritätsbedingung verletzt wird, erfolgt eine Meldung, in der auf diesen Namen Bezug genommen wird.

Wenn ein Benutzer eine neue Integritätsbedingung definiert, prüft SESAM/SQL, ob die Integritätsbedingung nicht durch schon bestehende Daten in der Datenbank verletzt wird. In diesem Fall wird die Definition der Integritätsbedingung abgewiesen. Ist die Tabelle leer, dann ist die Integritätsbedingung stets wahr.

SESAM/SQL kennt unterschiedliche Integritätsbedingungen:

Nicht-NULL-Bedingung

Die Nicht-NULL-Bedingung (NOT NULL) fordert, dass eine Spalte keine NULL-Werte enthalten darf. Sie kann nur als Spaltenbedingung angegeben werden.

Eindeutigkeitsbedingung

Die Eindeutigkeitsbedingung (UNIQUE) für eine Spalte fordert, dass in der angegebenen Spalte jeder vom NULL-Wert verschiedene Wert nur einmal vorkommen darf. Die Eindeutigkeitsbedingung für eine Spaltenkombination fordert, dass in der angegebenen Spaltenkombination jede Wertekombination, die keinen NULL-Wert enthält, nur einmal vorkommen darf.

Primärschlüsselbedingung

Die Primärschlüsselbedingung (PRIMARY KEY) legt eine Spalte oder eine Spaltenkombination als Primärschlüssel einer Tabelle fest. Die Primärschlüsselbedingung fordert, dass für die Spalte bzw. die Spaltenkombination die Eindeutigkeitsbedingung und die Nicht-NULL-Bedingung erfüllt sind.

Eine Tabelle darf nur einen Primärschlüssel besitzen. Der Primärschlüssel darf nicht vom Datentyp VARCHAR oder NVARCHAR sein.

Für CALL-DML-Tabellen darf nur die Primärschlüsselbedingung definiert werden.

Referenzbedingung

Die Referenzbedingung ([FOREIGN KEY]...REFERENCES) legt eine Spalte oder Spaltenkombination als Fremdschlüssel einer Tabelle fest. Der Fremdschlüssel bezieht sich auf eine oder mehrere Spalten einer anderen Tabelle. Für diese Spalten muss eine Eindeutigkeitsbedingung gelten. Die Tabelle, die den Fremdschlüssel enthält, wird **referenzierende** Tabelle genannt, die Tabelle, für deren Spalten die Eindeutigkeitsbedingung gelten muss, wird **referenzierte** Tabelle genannt. Anzahl und Datentypen der einander zugeordneten Spalten der referenzierenden und der referenzierten Tabelle müssen gleich sein. Bei der referenzierenden und der referenzierten Tabelle kann es sich auch um dieselbe Basistabelle handeln.

Ein Satz der referenzierenden Tabelle erfüllt die Referenzbedingung genau dann, wenn er entweder in einer der referenzierenden Spalten den NULL-Wert enthält, oder diese Werte alle ungleich NULL sind und es einen Satz der referenzierten Tabelle gibt, der in den referenzierten Spalten dieselben Werte enthält. Ist beim Einfügen eines Satzes oder Ändern von Spaltenwerten in der referenzierenden Tabelle oder beim Löschen und Ändern von Sätzen der referenzierten Tabelle die Referenzbedingung nicht erfüllt, dann weist SESAM/SQL diese Tabellenoperation ab.

Bei einspaltigen Fremdschlüsseln fordert die Referenzbedingung, dass jeder vom NULL-Wert verschiedene Wert des Fremdschlüssels einer Tabelle als Wert einer bestimmten Spalte in einer anderen Tabelle vorkommt, die der Eindeutigkeitsbedingung genügt.

Bei mehrspaltigen Fremdschlüsseln muss jede darin vorkommende Wertekombination, die keinen NULL-Wert enthält, in der entsprechenden Spaltenkombination der referenzierten Tabelle auftreten. Diese Spaltenkombination muss der Eindeutigkeitsbedingung genügen. In SQL genügt somit ein Satz schon der Referenzbedingung, wenn er in mindestens einer Spalte des mehrspaltigen Fremdschlüssels einen NULL-Wert enthält.

Werden nach REFERENCES keine Spalte bzw. keine Spalten für die referenzierte Tabelle angegeben, dann wird der Primärschlüssel der referenzierten Tabelle verwendet.

Check-Bedingung

Die CHECK-Bedingung fordert, dass jeder Wert einer Spalte bzw. jede Wertekombination von Spalten eine Suchbedingung erfüllt. Die Suchbedingung darf sich nur auf die Tabelle beziehen, zu der die Spalte bzw. die Spalten gehören und keine Unterabfrage oder Transliteration zwischen EBCDIC und Unicode enthalten. Außerdem darf die Suchbedingung keine Umwandlung von Großbuchstaben in Kleinbuchstaben oder von Kleinbuchstaben in Großbuchstaben enthalten, wenn die umzuwandelnde Zeichenkette eine Unicode-Zeichenkette ist. Da Integritätsbedingungen nicht von einer bestimmten Anwendung abhängig sein dürfen, darf keine Benutzervariable, keine Zeitfunktion und kein Spezial-Literal in der Suchbedingung angegeben werden. Die Suchbedingung darf keine multiple Spalte referenzieren.

Löschen von Integritätsbedingungen

Eine Primärschlüsselbedingung kann nur gelöscht werden, indem die betreffende Tabelle mit DROP TABLE gelöscht wird. Die übrigen Integritätsbedingungen können mit ALTER TABLE DROP CONSTRAINT gelöscht werden oder implizit beim Löschen einer Tabelle mit DROP TABLE.

Eine Eindeutigkeitsbedingung kann mit ALTER TABLE DROP CONSTRAINT RESTRICT nur dann gelöscht werden, wenn sie nicht die Eindeutigkeitsbedingung für die referenzierten Spalten einer Referenzbedingung ist.

4.1.7 Index

Ein Index auf eine Basistabelle wird verwendet, um den Zugriff auf eine Tabelle zu beschleunigen. Ein Index ist eine baumartige Zugriffsstruktur, die einer Spalte oder Spaltenkombination einer bestimmten Tabelle zugeordnet ist und Verweise auf die Sätze dieser Tabelle enthält. Ein Index ordnet über eine sogenannte invertierte Liste jedem Wert einer Spalte die Sätze zu, die diesen Wert in der betreffenden Spalte enthalten. Entsprechendes gilt für Spaltenkombinationen.

Je nachdem, ob sich ein Index auf eine oder mehrere Spalten bezieht, spricht man von einem einfachen oder von einem zusammengesetzten Index.

SESAM/SQL verwendet einen Index,

- um schnell auf Sätze mit bestimmten Werten in den Indexspalten zugreifen zu können
- um die Sätze einer Tabelle in der Reihenfolge der Werte der Indexspalten zu liefern
- um Integritätsbedingungen auf einer oder mehreren Spalten des Index ohne Zugriff auf die Basistabelle auszuwerten.

Die zusätzliche Verwaltung eines Index erhöht den Aufwand bei Einfüge- und Änderungsanweisungen sowie bei Recovery-Maßnahmen. Daher sollten die folgenden Richtlinien berücksichtigt werden:

- Für Tabellen mit wenig Sätzen sollten keine Indizes erzeugt werden. Der Zeitgewinn durch einen Index wird bei kleinen Tabellen schon durch die Zeit für das Öffnen und Durchsuchen der Indexdatei zunichte gemacht.
- Für eine Spalte, in der nur sehr wenige unterschiedliche Werte vorkommen, sollte kein Index erzeugt werden.
- Bei Tabellen, die vorwiegend für die Datenwiedergewinnung verwendet werden, sollten Indizes häufiger verwendet werden als bei Tabellen, in denen oft geändert wird.

Ein Index wird mit den SQL-Anweisungen CREATE INDEX und ALTER TABLE erzeugt und mit DROP INDEX gelöscht.

Speicherstruktur von Indizes

Wenn ein Index mit CREATE INDEX angelegt wird, dann wird Speicherplatz, der sogenannte zusammenhängende Bereich des Index, reserviert. Die einzufügenden Werte werden in diesem Bereich gespeichert. Wenn ein Wert eingefügt werden soll und der Platz in diesem Bereich nicht mehr ausreicht, dann wird eine sogenannte Auslagerung, ein freier Block, angelegt. Dieser Block gehört logisch zum Index, liegt aber physikalisch nicht mehr im zusammenhängenden Bereich des Index.
Bei der nächsten Reorganisation des Anwender-Space mit der Anweisung REORG SPACE werden alle existierenden Tabellen und Indizes auf dem Anwender-Space neu aufgebaut. Damit verschwinden auch die Auslagerungen.

Index für eine Eindeutigkeitsbedingung

Für jede Spalte oder Spaltenkombination, für die die Eindeutigkeitsbedingung definiert ist, benötigt SESAM/SQL einen Index. Wenn bereits ein Index mit CREATE INDEX für die betreffende Spalte oder Spaltenkombination erzeugt wurde, so wird dieser Index zusätzlich für die Eindeutigkeitsbedingung verwendet.

Ansonsten erzeugt SESAM/SQL den Index automatisch. Der Name des so erzeugten Index beginnt mit UI, gefolgt von einer 16-stelligen Zahl.

REORG STATISTICS-Anweisung

REORG STATISTICS baut die globale Statistik für einen Index neu auf. REORG STATISTICS ist sinnvoll nach einer umfangreichen Änderung und Neuaufnahme. Eine aktualisierte Statistik-Information ermöglicht intern eine genauere Kostenabschätzung und Entscheidung für die Auswahl des günstigsten Zugriffsplans (siehe Seite 40).

4.1.8 Routine

SESAM/SQL unterscheidet folgende Routinen:

- **Prozeduren (Stored Procedures)**
- **User Defined Functions (UDF)**.

In SESAM/SQL wird der Oberbegriff **Routine** für Prozeduren und User Defined Functions (UDF) verwendet, wenn die Informationen sowohl für Prozeduren als auch für UDFs gelten.

Der Oberbegriff „SQL-invoked routine“ der SQL-Norm wird in SESAM/SQL nicht verwendet.

In einer Routine werden Abläufe von SQL-Anweisungen in der Datenbank gespeichert, die später mit einem einzigen Aufruf ausgeführt werden können. Eine Routine ist vergleichbar mit einem Unterprogramm, das vollständig, also ohne Datenaustausch mit dem Anwendungsprogramm, im DBH abläuft.

Der Text einer Routine ist in SESAM/SQL vollständig in der Programmiersprache SQL geschrieben.

Neben den üblichen DML-Anweisungen kann eine Routine auch lokale Definitions-, Kontroll- und Diagnoseanweisungen enthalten. Definitionsanweisungen definieren lokale Daten und Cursor und legen spezielle Fehlerbehandlungen fest. Kontrollanweisungen steuern den Ablauf der Routine, z.B. durch Laufschleifen oder Bedingungen. Diagnoseanweisungen stellen Informationen zu einem möglicherweise fehlerhaften Verlauf der Routine bereit.

Zur Ausführung einer Routine wird das EXECUTE-Privileg benötigt.

Informationen über Routinen werden in den Informationsschemata bereitgestellt.

Prozeduren können mit Ein- und Ausgabeparametern definiert werden. Eingabeparameter werden beim Prozeduraufruf durch Argumente versorgt. Ausgabewerte werden von der Prozedur an vorgegebenen Ablageorten abgelegt.
Eine Prozedur wird mit CREATE PROCEDURE erzeugt, mit der SQL-Anweisung CALL ausgeführt und mit DROP PROCEDURE gelöscht.

UDFs können mit Eingabeparametern definiert werden. Eingabeparameter werden beim Funktionsaufruf in einem Ausdruck durch Argumente versorgt. UDFs haben genau einen Rückgabewert (SQL-Anweisung RETURN).
Eine UDF wird mit CREATE FUNCTION erzeugt, durch einen Funktionsaufruf ausgeführt und mit DROP FUNCTION gelöscht.

4.1.9 BLOB-Konstrukte

Mit SESAM/SQL können Sie Multimedia-Dateninhalte persistent und ausfallsicher in einer Datenbank abspeichern.

Für Multimedia-Dateninhalte werden im Wesentlichen vier große Bereiche unterschieden:

- Text
- Grafik
- Audio (Sound)
- Video

Diese großen Datenmengen werden als BLOBs (**B**inary **L**arge **Ob**jects) bezeichnet. Exakt definiert sind BLOBs Folgen von Bytes variabler Länge, die bis zu 2^{31}-1 Bytes groß sein können. SESAM/SQL speichert diese BLOBs als BLOB-Objekte. Dabei gelten für BLOB-Objekte die in SESAM/SQL etablierten Sicherheitsmechanismen.

Bei der Handhabung von BLOB-Objekten wird unterschieden in:

- Verwaltung der BLOB-Objekte, diese erfolgt mit SESAM/SQL
- Inhaltliche Bearbeitung der BLOB-Objekte, diese erfolgt außerhalb von SESAM/SQL mit Programmen, die auf den verwendeten Dateninhalt zugeschnitten sind (Beispiel: Bearbeitung von Bildern mit Bildbearbeitungsprogrammen beliebiger Hersteller).

Zur Arbeit mit BLOB-Objekten werden in SESAM/SQL folgende Elemente genutzt. Ihr Zusammenspiel ist in Bild 8 auf Seite 100 grafisch dargestellt.

BLOB-Objekte	In SESAM/SQL werden BLOBs als BLOB-Objekte bezeichnet, da sie nicht nur aus einem Wert sondern auch aus zugeordneten Attributen bestehen.
BLOB-Tabellen	Dies sind besondere Basistabellen, die als Speicherort für BLOB-Objekte dienen. Ihr Aufbau ist auf die Speicherung von BLOB-Objekten abgestimmt. Die BLOB-Objekte einer Tabelle werden als Klasse bezeichnet.
BLOB-Werte	Der Wert eines BLOB-Objekts wird als BLOB-Wert bezeichnet.
Attribute	Ein BLOB-Objekt besitzt mehrere Attribute. Sie geben Auskunft über das Entstehungsdatum und das letzte Änderungsdatum und enthalten weitere Angaben, die das Objekt beschreiben. Um die einzelnen Attribute unterscheiden zu können, werden sie mit Namen (Tags) ausgezeichnet
REF-Werte	Jedes BLOB-Objekt und jede Klasse besitzt einen eindeutigen REF-Wert, der das Objekt / die Klasse referenziert. SESAM/SQL benutzt diesen REF-Wert, um auf das BLOB-Objekt / die Klasse zuzugreifen.

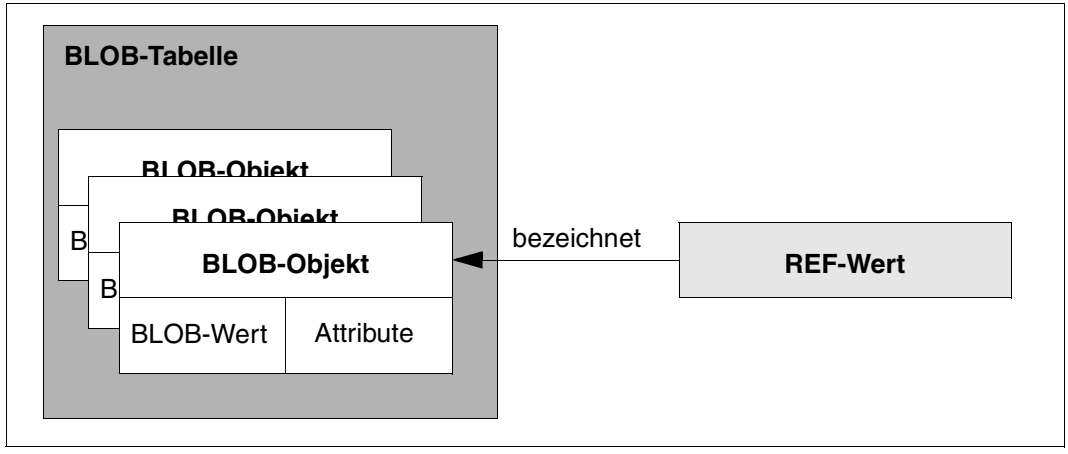


Bild 8: Struktur der BLOB-Konstrukte

Wenn Sie ein BLOB-Objekt erzeugen, so speichert SESAM/SQL seinen BLOB-Wert und die ihm zugeordneten Attribute in einer BLOB-Tabelle. Der BLOB-Wert wird dabei stückweise in mehrere Zeilen der BLOB-Tabelle geschrieben. Diese Speichermethode ermöglicht einen effizienten sequenziellen Zugriff auf die BLOB-Werte.

Um BLOB-Objekte in beliebige Basistabellen einzubinden, erzeugt SESAM/SQL zu jedem BLOB-Objekt einen eindeutigen REF-Wert. Dieser bezeichnet das BLOB-Objekt, solange es existiert. Diesen REF-Wert können Sie in REF-Spalten beliebiger Basistabellen speichern (siehe Abschnitt „Spalte“ auf Seite 92).

Die BLOB-Objekte einer BLOB-Tabelle bilden die Objekte einer Klasse. Dementsprechend gibt es auch einen REF-Wert der Klasse, der alle BLOB-Objekte einer BLOB-Tabelle bezeichnet. BLOB-Objekte, Klassen und Attribute von BLOB-Objekten sowie BLOB-Werte müssen Sie mit Hilfe der Schnittstelle SESAM-CLI (**C**all **L**evel **I**nterface) ansprechen.

4.1.9.1 BLOB-Tabellen

BLOB-Tabellen erzeugen Sie mit der Anweisung CREATE TABLE OF BLOB. Sie dienen als Speicherort von BLOB-Objekten. SESAM/SQL behandelt BLOB-Tabellen grundsätzlich wie andere SQL-Tabellen. Sie werden als „normale“ Basistabellen in den Informationsschemata angezeigt. Bei View- und Indexdefinitionen bestehen keine Unterschiede zu anderen SQL-Tabellen.

Die BLOB-Werte des BLOB-Objekts werden mit Aufrufen des SESAM-CLI stückweise in mehreren Zeilen der Tabelle gespeichert. SESAM/SQL verwendet dabei eine Stückgröße von 31000 Byte. Diese Größe nutzt die Bandbreite bei der Übertragung des BLOB-Werts zwischen SQL-Client und SQL-Server optimal aus.

SESAM/SQL legt die Struktur der BLOB-Tabelle bei ihrer Erzeugung fest. Eine Spaltendefinition durch den Anwender ist deshalb an dieser Stelle nicht möglich.

Eine BLOB-Tabelle besitzt die folgenden Spalten:

- Die Spalte OBJ_NR ist vom Datentyp INTEGER und enthält die laufende Nummer des BLOB-Objekts innerhalb der Tabelle.
- Die Spalte SLICE_NR ist vom Datentyp INTEGER und enthält die laufende Nummer des Teilstücks des BLOB-Werts.
- Die Spalte SLICE_VAL ist vom Typ VARCHAR(31000). Sie enthält jeweils die Teilstücke des BLOB-Werts. Die Einträge ab der Teilstücknummer 1 enthalten den Wert in 31000 Byte großen Teilstücken. Das letzte Teilstück kann natürlich kleiner sein. In der Zeile mit der Teilstücknummer 0 sind Verwaltungsinformationen für das BLOB-Objekt gespeichert. Die Voreinstellungen für diese Spalte sind die in der OF BLOB-Klausel vom Anwender definierten Attribute wie MIME und USAGE. Zusätzlich enthalten die Voreinstellungen die Attribute CREATED und UPDATED. Diese Attribute geben Auskunft über das Entstehungs- und letzte Änderungsdatum des BLOB-Objekts.
- Die Spalte OBJ_REF ist vom Typ CHAR(237). Bei Teilstücknummer 0 enthält diese Spalte den REF-Wert des BLOB-Objekts. Bei allen anderen Teilstücken ist der Spaltenwert NULL. Als Voreinstellung erhält die Spalte den REF-Wert der Klasse dieser Tabelle. Die Spalte ist mit der Bedingung UNIQUE definiert.

Die Spalten OBJ_NR und SLICE_NR bilden den Primärschlüssel einer BLOB-Tabelle. Für diese Primärschlüsselbedingung vergibt SESAM/SQL hier wie üblich intern generierte Nummern, die im selben Schema nicht mehr genutzt werden können.

Es ist möglich, weitere Spalten mit ALTER TABLE anzufügen. Mit ALTER TABLE können Sie auch BLOB-Tabellen ändern. Diese Änderungen können jedoch dazu führen, dass die BLOB-Tabelle nicht mehr durch CLI-Aufrufe angesprochen werden kann, da die CLI-Aufrufe an die von SESAM/SQL vorgegebene Struktur angepasst sind.

BLOB-Tabellen können mit DROP TABLE gelöscht werden. Dieser Aufruf löscht auch alle enthaltenen BLOB-Objekte.

4.1.9.2 BLOB-Objekte

Ein BLOB-Objekt besitzt einen BLOB-Wert und verschiedene Attribute.

Die Speicherung von BLOB-Werten in „normalen Tabellen“ erfolgt über die REF-Werte. Eine direkte Speicherung ist nicht möglich. Dies hat den Vorteil, dass Sie alle üblichen SQL-Operationen mit REF-Werten in Basistabellen ausführen können. Mit BLOB-Werten wäre dies nicht möglich.

Die Attribute eines BLOB-Objekts erhalten bei dessen Erzeugung zunächst die Voreinstellungen der Klasse. Die Voreinstellungen der Klasse werden mit *mimeklausel*, *usageklausel* und *alphanumerisches_literal* festgelegt, wenn die BLOB-Tabelle erzeugt wird.

Diese Attribute werden mit den Namen (Tags) MIME und USAGE ausgezeichnet. Neben diesen optionalen Attributen werden der Entstehungszeitpunkt und das letzte Änderungsdatum mit den Tags CREATED und UPDATED festgehalten. Die Attribute MIME und USAGE können Sie mit dem CLI-Aufruf SQL_BLOB_TAG_PUT ersetzen.

BLOB-Objekte und -Werte erzeugen, lesen oder setzen Sie mit Aufrufen des SESAM-CLI. Die inhaltliche Bearbeitung der BLOB-Werte erfolgt dagegen nicht in SESAM/SQL sondern in den objektspezifischen Programmen. Für den Transfer der BLOB-Werte aus SESAM/SQL in ein anderes System, zum Beispiel in BS2000, stehen zwei verschiedene Möglichkeiten zur Verfügung:

- Der BLOB-Wert kann mit SQL_BLOB_VAL_GET in einem einzigen Schritt ausgelesen werden.
- Mit der Befehlsfolge SQL_BLOB_VAL_OPEN, SQL_BLOB_VAL_FETCH und SQL_BLOB_VAL_CLOSE kann der BLOB-Wert stückweise ausgelesen werden (siehe Abschnitt „SESAM-CLI“ auf Seite 104).

Nicht mehr benötigte BLOB-Objekte sollten Sie aufgrund ihrer Größe löschen.

4.1.9.3 REF-Werte

Bei der Erzeugung eines neuen BLOB-Objekts generiert SESAM/SQL einen REF-Wert. Diesen Wert können Sie in Spalten beliebiger Tabellen verwenden, um das BLOB-Objekt zu referenzieren. Dazu werden REF-Spalten in Basistabellen definiert. Die Syntax der Spaltendefinition ist in Abschnitt „Spalte“ auf Seite 92 beschrieben.

Die REF-Spalte erhält neben dem Datentyp CHAR(237) als Defaultwert den REF-Wert der Klasse. Bestimmte Aufrufe des SESAM-CLI, wie zum Beispiel das Erzeugen eines neuen BLOB-Objekts, können den REF-Wert der Klasse erfordern.

Ein REF-Wert hat grundsätzlich folgende Struktur: *ss*/*tt*?UID=*uuuu*&OID=*nn*

ss ist der einfache Schemaname der BLOB-Tabelle ohne Datenbanknamen.

tt ist der einfache Tabellenname der BLOB-Tabelle ohne Schema- und Datenbanknamen.

uuuu ist eine eindeutige Identifikationsnummer des BLOB-Objekts, die aus 32 Hexadezimalziffern besteht. Beim REF-Wert der Klasse sind alle Ziffern 0.

nn ist die Nummer des BLOB-Objekts in der BLOB-Tabelle. Beim REF-Wert der Klasse ist diese Nummer 0.

Zwei REF-Werte sind genau dann gleich, wenn sie dasselbe BLOB-Objekt bezeichnen. Sind zwei REF-Werte unterschiedlich, so sind auch die referenzierten BLOB-Objekte unterschiedlich. Deren BLOB-Werte können hingegen gleich sein. Wird ein BLOB-Objekt gelöscht, bezeichnet sein zugehöriger REF-Wert nie wieder ein BLOB-Objekt.

4.1.9.4 SESAM-CLI

BLOB-Objekte, ihre Werte und Attribute sowie die Klassen von BLOB-Objekten werden mit Aufrufen der Schnittstelle SESAM-CLI (**C**all **L**evel **I**nterface) angesprochen. CLI-Aufrufe erfolgen aus C- oder COBOL-Programmen.

Im Folgenden finden Sie eine Übersicht über die Aufrufe des SESAM-CLI und ihrer jeweiligen Funktion. Die einzelnen Aufrufe werden ausführlich im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“ beschrieben.

Operationen mit Klassen von BLOB-Objekten

CLI-Aufruf	Kurzform	Funktion
SQL_BLOB_CLS_REF	SQLBCRE	REF-Wert der Klasse ausgeben
SQL_BLOB_CLS_ISBTAB	SQLBCIS	Prüfung, ob BLOB-Tabelle vorliegt

Tabelle 16: CLI-Aufrufe für Operationen mit Klassen von BLOB-Objekten

BLOB-Objekte erzeugen und löschen

CLI-Aufruf	Kurzform	Funktion
SQL_BLOB_OBJ_CREATE	SQLBOCR	Objekt erzeugen (Objektnummer sequeziell)
SQL_BLOB_OBJ_CREAT2	SQLBOC2	Objekt erzeugen (Objektnummer bereichspezifisch)
SQL_BLOB_OBJ_DROP	SQLBODR	Objekt löschen

Tabelle 17: CLI-Aufrufe für BLOB-Objekte

Attribut eines BLOB-Objekts lesen und setzen

CLI-Aufruf	Kurzform	Funktion
SQL_BLOB_TAG_GET	SQLBTGE	Attributwert lesen
SQL_BLOB_TAG_PUT	SQLBTPU	Attributwert setzen

Tabelle 18: CLI-Aufrufe für Attribute von BLOB-Objekten

BLOB-Werte lesen und schreiben

CLI-Aufruf	Kurzform	Funktion
SQL_BLOB_VAL_GET	SQLBVGE	BLOB-Wert lesen
SQL_BLOB_VAL_PUT	SQLBVPU	BLOB-Wert setzen
SQL_BLOB_VAL_LEN	SQLBVLE	Länge des BLOB-Werts ausgeben

Tabelle 19: CLI-Aufrufe für BLOB-Werte

Sequenzielle Bearbeitung von BLOB-Werten

CLI-Aufruf	Kurzform	Funktion
SQL_BLOB_VAL_OPEN	SQLbvop	Access-Handle öffnen
SQL_BLOB_VAL_CLOSE	SQLbvcl	Access-Handle schließen
SQL_BLOB_VAL_FETCH	SQLbvfe	Sequenzielles Lesen eines BLOB-Werts
SQL_BLOB_VAL_STOW	SQLbvst	Sequenzielles Setzen eines BLOB-Werts

Tabelle 20: CLI-Aufruf für einzelne Sequenzen der BLOB-Werte

4.2 SQL-Anweisungen

SQL-Anweisungen können nach Gesichtspunkten klassifiziert werden, z.B.:

- ob sie eine Transaktion einleiten (siehe Abschnitt „SQL-Transaktion“ auf Seite 134)
- ob sie ausführbare SQL-Anweisungen sind (siehe Seite 138)
- ob sie dynamisch übersetzbar sind (siehe Seite 142)

SQL-Anweisungen können auch nach ihrer Funktion klassifiziert werden. Die SQL-Norm unterscheidet folgende Klassen von Anweisungen:

- SQL-Anweisungen zur Schemadefinition und -verwaltung
- SQL-Anweisungen zum Abfragen und Ändern von Daten
- SQL-Anweisungen zur Gestaltung und Verwaltung von Routinen
- SQL-Anweisungen zur Transaktionsverwaltung
- SQL-Anweisungen zur Sessionsteuerung
- SQL-Anweisungen der dynamischen SQL
- WHENEVER-Anweisung zur ESQL-Fehlerbehandlung

SESAM/SQL kennt zusätzlich die folgenden Klassen von SQL-Anweisungen:

- SQL-Anweisungen zur Verwaltung der Speicherstruktur
- SQL-Anweisungen zur Verwaltung von Benutzereinträgen
- Utility-Anweisungen

Die SQL-Anweisungen zur Schemadefinition und -verwaltung werden auch als DDL-Anweisungen („Data Definition Language“) bezeichnet und die SQL-Anweisungen zum Abfragen und Ändern von Daten werden als DML-Anweisungen („Data Manipulation Language“) bezeichnet.

Im Folgenden werden die SQL-Anweisungen für jede der genannten Klassen zusammengestellt und jeweils kurz beschrieben. SESAM/SQL-spezifische Erweiterungen sind grau hinterlegt. Auf die Funktionalität der SQL-Anweisungen wurde im Abschnitt „SQL-Objekte einer SESAM/SQL-Datenbank“ auf Seite 79 eingegangen.

4.2.1 SQL-Anweisungen zur Schemadefinition und -verwaltung

SQL-Anweisung	Funktion
CREATE SCHEMA	erzeugt ein Schema
DROP SCHEMA	löscht ein leeres Schema
CREATE TABLE	erzeugt eine Basistabelle
ALTER TABLE	ändert Spalten, Indizes und Integritätsbedingungen einer Tabelle
DROP TABLE	löscht eine Basistabelle und die zugehörigen Indizes
CREATE VIEW	erzeugt einen View
DROP VIEW	löscht die Definition eines View
GRANT	vergibt an einen Benutzer Tabellen- oder Sonderprivilegien
REVOKE	entzieht einem Benutzer Tabellen- oder Sonderprivilegien

Tabelle 21: SQL-Anweisungen zur Schemadefinition und -verwaltung

4.2.2 SQL-Anweisungen zum Abfragen und Ändern von Daten

SQL-Anweisung	Funktion
SELECT[1]	überträgt Werte eines Satzes in eine Benutzervariable
UPDATE	ändert Spaltenwerte von Sätzen gemäß einer Suchbedingung
DELETE	löscht Sätze in einer Tabelle gemäß einer Suchbedingung
INSERT	fügt Zeilen in eine Tabelle ein
MERGE	fügt neue Sätze in eine Tabelle ein und ändert vorhandene Sätze
CALL	führt eine Prozedur aus
DECLARE CURSOR[2]	vereinbart einen Cursor
OPEN[2]	öffnet einen Cursor
FETCH[2]	positioniert auf einen Satz der Cursortabelle und liest ihn
UPDATE...CURRENT[2]	ändert den aktuellen Satz der Cursortabelle
DELETE... CURRENT[2]	löscht den aktuellen Satz der Cursortabelle
CLOSE[2]	schließt einen Cursor
STORE[2]	speichert die aktuelle Cursorposition
RESTORE[2]	stellt einen mit STORE gespeicherten Cursor wieder her

Tabelle 22: SQL-Anweisungen zum Abfragen und Ändern von Daten

[1] Für die SELECT-Anweisung gibt es eine dynamisch übersetzbare Form.

[2] Diese Anweisungen gibt es auch für einen dynamischen Cursor.

4.2.3 SQL-Anweisungen zur Gestaltung und Verwaltung von Routinen

SQL-Anweisung	Funktion
`CREATE PROCEDURE`	erzeugt eine Prozedur
`CREATE FUNCTION`	erzeugt eine User Defined Function (UDF)
COMPOUND-Anweisung, `CASE, FOR, IF, ITERATE, LEAVE, LOOP, REPEAT, RETURN, SET, WHILE`	Kontrollanweisungen für eine Routine (in der CREATE PROCEDURE- oder CREATE FUNCTION-Anweisung)
`GET DIAGNOSTICS; SIGNAL; RESIGNAL`	Diagnoseanweisungen für eine Routine (in der CREATE PROCEDURE- oder CREATE FUNCTION-Anweisung)
`DROP PROCEDURE`	löscht eine Prozedur
`DROP Function`	löscht eine User Defined Function (UDF)

Tabelle 23: SQL-Anweisungen zur Gestaltung und Verwaltung von Routinen

4.2.4 SQL-Anweisungen zur Transaktionsverwaltung

SQL-Anweisung	Funktion
`SET TRANSACTION`	legt den Isolationslevel und den Transaktionsmodus fest
`COMMIT WORK`	schreibt Änderungen einer Transaktion fest
`ROLLBACK WORK`	macht Änderungen einer Transaktion rückgängig

Tabelle 24: SQL-Anweisungen zur Transaktionsverwaltung

4.2.5 SQL-Anweisungen zur Session-Steuerung

SQL-Anweisung	Funktion
SET CATALOG	legt den voreingestellten Datenbanknamen für dynamisch übersetzbare SQL-Anweisungen und Cursorbeschreibungen fest
SET SCHEMA	legt den voreingestellten Schemanamen für dynamisch übersetzbare SQL-Anweisungen und Cursorbeschreibungen fest
SET SESSION AUTHORIZATION	legt einen Berechtigungsschlüssel für die aktuelle SQL-Session fest
PERMIT	hat keine Wirkung; nur wegen Kompatibilität zu SESAM/SQL V1 vorhanden

Tabelle 25: SQL-Anweisungen zur Sessionsteuerung

4.2.6 SQL-Anweisungen der dynamischen SQL

SQL-Anweisung	Funktion
PREPARE	prüft eine dynamisch übersetzbare SQL-Anweisung oder Cursorbeschreibung und bereitet sie für die Ausführung vor
EXECUTE	führt eine mit PREPARE vorbereitete Anweisung aus
EXECUTE IMMEDIATE	prüft eine dynamisch übersetzbare Anweisung und führt sie aus
ALLOCATE DESCRIPTOR	legt einen SQL-Deskriptorbereich an
GET DESCRIPTOR	liest Einträge aus einem SQL-Deskriptorbereich
SET DESCRIPTOR	ändert einen SQL-Deskriptorbereich
DEALLOCATE DESCRIPTOR	gibt einen SQL-Deskriptorbereich frei
DESCRIBE	informiert über Ergebnisspalten oder Eingabewerte einer dynamisch übersetzten Anweisung oder Cursorbeschreibung

Tabelle 26: SQL-Anweisungen der dynamischen SQL

4.2.7 WHENEVER-Anweisung zur ESQL-Fehlerbehandlung

SQL-Anweisung	Funktion
WHENEVER	legt die Reaktion auf eine nicht erfolgreich durchgeführte SQL-Anweisung fest

Tabelle 27: WHENEVER-Anweisung zur ESQL-Fehlerbehandlung

4.2.8 SQL-Anweisungen zur Verwaltung der Speicherstruktur

SQL-Anweisung	Funktion
CREATE STOGROUP	erzeugt eine neue Storage Group
ALTER STOGROUP	ändert die Definition einer Storage Group
DROP STOGROUP	löscht eine Storage Group
CREATE SPACE	erzeugt einen Space und trägt Spacenamen in Metadaten ein
ALTER SPACE	ändert die Eigenschaften für einen Space
DROP SPACE	löscht einen Anwender-Space
CREATE INDEX	erzeugt einen Index für eine Basistabelle
DROP INDEX	löscht einen Index
REORG STATISTICS	erzeugt eine globale Statistik für einen Index neu

Tabelle 28: SQL-Anweisungen zur Verwaltung der Speicherstruktur

4.2.9 SQL-Anweisungen zur Verwaltung von Benutzereinträgen

SQL-Anweisung	Funktion
CREATE USER	erzeugt einen neuen Berechtigungsschlüssel
DROP USER	löscht einen Berechtigungsschlüssel und die zugehörigen Systemzugänge
CREATE SYSTEM_USER	ordnet Systembenutzern Berechtigungsschlüssel für eine Datenbank zu
DROP SYSTEM_USER	löscht die Zuordnung eines Berechtigungsschlüssels zu einem Systembenutzer

Tabelle 29: SQL-Anweisungen zur Verwaltung von Benutzereinträgen

4.2.10 Utility-Anweisungen

Utility-Anweisungen sind Anweisungen in SQL-Syntax, die Utility-Funktionen für die Datenbankverwaltung bereitstellen (siehe Kapitel „Utility-Konzept“ auf Seite 159). Es handelt sich um SESAM/SQL-spezifische Erweiterungen. Im Unterschied zu SQL-Anweisungen können Utility-Anweisungen nur außerhalb einer Transaktion gegeben werden, d.h., es darf keine Transaktion mehr offen sein, wenn eine Utility-Anweisung eingegeben wird. Eine Utility-Anweisung wird über eine Folge interner Transaktionen abgewickelt und ist deshalb nicht rücksetzbar.

4.3 Grundlegende SQL-Sprachmittel

Dieser Abschnitt beschreibt grundlegende Sprachmittel von SESAM/SQL, die in SQL-Anweisungen verwendet werden.

Im Einzelnen behandelt der Abschnitt:

- Namen für SQL-Objekte
- Datentypen
- Werte
- Ausdrücke
- Funktionen
- CASE-Ausdruck
- CAST-Ausdruck
- Suchbedingung
- Prädikate
- Abfrage-Ausdruck
- SELECT-Ausdruck
- Unterabfrage
- Join-Ausdruck

4.3.1 Namen

Namen sind Zeichenfolgen, die verwendet werden, um SQL-Objekte zu identifizieren.

Der Name eines Objekts wird in der Regel bei der Definition des Objekts mit der entsprechenden SQL-Anweisung festgelegt. Damit ist der Name eingeführt und das Objekt kann in nachfolgenden Anweisungen mit diesem Namen angesprochen werden.

SESAM/SQL unterscheidet einfache Namen und qualifizierte Namen.

Einfache Namen

Einfache Namen sind entweder reguläre Namen, die nicht in Anführungszeichen eingeschlossen werden, oder Spezialnamen, die in Anführungszeichen eingeschlossen werden müssen.

Reguläre Namen

Reguläre Namen setzen sich aus Buchstaben, Ziffern und dem Unterstrich (_) zusammen. Das erste Zeichen eines regulären Namens muss ein Buchstabe sein. Bei der Interpretation werden Kleinbuchstaben in Großbuchstaben umgewandelt. Für reguläre Namen dürfen keine reservierten SQL-Schlüsselwörter verwendet werden.

Spezialnamen

Spezialnamen dürfen dagegen auch reservierte SQL-Schlüsselwörter sein und Zeichen enthalten, die für reguläre Namen nicht erlaubt sind. Groß- und Kleinbuchstaben werden unterschieden. Spezialnamen werden durch Anführungszeichen begrenzt. Das erste Zeichen nach dem einleitenden Anführungszeichen darf kein Unterstrich (_) sein. Ein Anführungszeichen innerhalb eines Spezialnamens muss doppelt angegeben werden. Das doppelte Anführungszeichen zählt dabei als ein Zeichen.

Leerzeichen am Ende von Spezialnamen sind nicht signifikant. Z.B. sind die folgenden Schreibweisen alle äquivalent:

KUNDE Kunde "KUNDE" "KUNDE "

Qualifizierte Namen

Um verschiedene Objekte gleichen Namens innerhalb der SQL-Anwendung eindeutig identifizieren zu können, gibt es die Möglichkeit, Namen zu qualifizieren. Namen für Objekte können in SQL explizit qualifiziert werden, indem den Namen, durch einen Punkt getrennt, ein Datenbank-, Schema- oder Tabellen- bzw. Korrelationsname vorangestellt wird.

Qualifiziert werden können die Namen für

- Schema, Space, Storage Group, Tabelle, Index. Integritätsbedingung und Routine mit dem Datenbanknamen
- Tabelle, Index, Integritätsbedingung und Routine mit dem Schemanamen
- Spalte mit dem Tabellen- bzw. Korrelationsnamen.

Namen für SQL-Objekte mit Ausnahme von Spalten müssen in der Regel explizit oder implizit qualifiziert werden. Ist keine explizite Qualifikation durch den Datenbank- bzw. Schemanamen angegeben, dann wird ein Name implizit mit dem voreingestellten Datenbank- bzw. voreingestellten Schemanamen qualifiziert. Die voreingestellten Datenbank- und Schemanamen werden mit der Precompiler-Option SOURCE-PROPERTIES (siehe Handbuch „ESQL-COBOL für SESAM/SQL-Server“) bzw. in den Konfigurationsdaten für den Utility-Monitor (siehe Handbuch „Utility-Monitor“) angegeben. Für dynamisch übersetzbare Anweisungen kann der Anwender mit SET CATALOG und SET SCHEMA voreingestellte Datenbank- bzw. Schemanamen angeben (siehe Seite 154).

Beispiel

Tabelle `KUNDE` im Schema `AUFTRAGSVER` der Datenbank `AUFTRAGKUNDEN`.

```
auftragkunden.auftragsver.kunde
```

4.3.2 Datentypen

Der Datentyp legt den Bereich zulässiger Werte einer Spalte fest. In SQL wird der Datentyp einer Spalte mit der Spaltendefinition (siehe Seite 92) in den SQL-Anweisungen CREATE TABLE oder ALTER TABLE festgelegt.

BLOBs (Binary Large Objects) basieren in SESAM/SQL auf vorhandenen Datentypen und sind daher kein neuer Datentyp. Die Struktur und Bearbeitung solcher Objekte wird im Abschnitt „BLOB-Konstrukte“ auf Seite 99 beschrieben.

Datentyp	SQL-Schreibweise	Wertebereich
alphanumerischer Datentyp: alphanumerische Zeichenkette fester Länge	CHAR[ACTER] [(*länge*)] *länge*: feste Länge der Spalte in Zeichen; vorzeichenlose Ganzzahl zwischen 1 und 256	alphanumerische Zeichenketten der Länge *länge* Der Datentyp CHAR wird nur in der signifikanten Länge ohne die nachfolgenden Leerzeichen abgespeichert.
alphanumerischer Datentyp: alphanumerische Zeichenketten der Längen 0 bis *max*	{ CHAR[ACTER] VARYING(*max*) | VARCHAR (*max*) } *max*: maximale Länge der Spalte in Zeichen; vorzeichenlose Ganzzahl zwischen 1 und 32000	alphanumerische Zeichenketten kleiner oder gleich der Länge *max*
National-Datentyp: National-Zeichenketten fester Länge	{ NATIONAL CHAR[ACTER] | NCHAR } [(*cu_länge* [CODE_UNITS])] *cu_länge:* feste Länge der Spalte in Code Units (1 Code Unit = 2 Byte); vorzeichenlose Ganzzahl zwischen 1 und 128	National-Zeichenketten der Länge *cu_länge* Der Datentyp NCHAR wird nur in der signifikanten Länge ohne die nachfolgenden Leerzeichen abgespeichert.
National-Datentyp: National-Zeichenketten der Länge 0 bis *cu_max*	{ NATIONAL CHAR[ACTER] VARYING | NCHAR VARYING | NVARCHAR } (*cu_max* [CODE_UNITS]) *cu_max*: maximale Länge der Spalte in Code Units (1 Code Unit = 2 Byte); vorzeichenlose Ganzzahl zwischen 1 und 16000	National-Zeichenketten kleiner oder gleich der Länge *cu_max*
numerisch: kleine Ganzzahl	SMALLINT	-2^{15} bis $2^{15}-1$

Tabelle 30: Datentypen in SESAM/SQL (Teil 1 von 2)

Datentyp	SQL-Schreibweise	Wertebereich
numerisch: große Ganzzahl	INT[EGER]	-2^{31} bis $2^{31}-1$
numerisch: Festpunktzahl	NUMERIC [(*stellen*[,*bruchteil*])] *stellen*: Anzahl der Dezimalstellen; vorzeichenlose Ganzzahl zwischen 1 und 31 *bruchteil*: Anzahl der Nachkommastellen; vorzeichenlose Ganzzahl zwischen 0 und *stellen*	Festpunktzahlen, deren Betrag 0 ist oder im Bereich $10^{-\text{bruchteil}}$ bis $10^{\text{stellen-bruchteil}}-10^{-\text{bruchteil}}$ liegt Die Datentypen NUMERIC und DECIMAL werden nur in der signifikanten Länge ohne die führenden Nullen abgespeichert.
numerisch: Festpunktzahl	DEC[IMAL] [(*stellen*[,*bruchteil*])] *stellen*: Anzahl der Dezimalstellen; vorzeichenlose Ganzzahl zwischen 1 und 31 *bruchteil*: Anzahl der Nachkommastellen; vorzeichenlose Ganzzahl zwischen 0 und *stellen*	
numerisch: Gleitpunktzahl - einfache Genauigkeit	REAL	Gleitpunktzahlen, deren Betrag 0 ist oder im Bereich $5.4E^{-79}$ bis $7.2E^{+75}$ liegt
numerisch: Gleitpunktzahl - doppelte Genauigkeit	DOUBLE PRECISION	
numerisch: Gleitpunktzahl	FLOAT[(*stellen*)] *stellen*: Mindestanzahl der Binärstellen für die Mantisse; vorzeichenlose Ganzzahl zwischen 1 und 53	
Zeit-Datentyp: Datum	DATE	Datumsangaben aus dem Bereich 0001-01-01 bis 9999-12-31
Zeit-Datentyp: Uhrzeit	TIME(3)	Uhrzeiten aus dem Bereich 00:00:00.000 bis 23:59:61.999
Zeit-Datentyp: Zeitstempel mit Datum und Uhrzeit	TIMESTAMP(3)	Datumsangaben wie bei DATE, Uhrzeit wie bei TIME(3)

Tabelle 30: Datentypen in SESAM/SQL (Teil 2 von 2)

4.3.3 Werte

Werte können in SQL-Anweisungen angegeben werden, um Spaltenwerte einzutragen oder zu ändern. Sie können in Ausdrücken durch Operatoren verknüpft und in Vergleichen verwendet werden. Sie können in Routinen auch als Parameter und lokale Variablen verwendet werden. Werte können als Literale direkt angegeben oder in Benutzervariablen (siehe Seite 139) übergeben werden. In dynamisch übersetzbaren Anweisungen (siehe Seite 142) kann das Zeichen „?“ als Platzhalter für einen Wert stehen. Jeder Wert hat einen Datentyp.

SESAM/SQL unterscheidet zwischen NULL-Werten und Nicht-NULL-Werten. Entsprechend den Datentypen unterscheidet man bei den Nicht-NULL-Werten zwischen Zeichenketten (alphanumerischen Werten und National-Werten), numerischen Werten und Zeitwerten.

Für jede dieser Gruppen gibt es entsprechende Formate, um Literale anzugeben.

REF-Werte, die im Zusammenhang mit BLOB (Binary Large Object) auftauchen, sind keine Werte im herkömmlichen Sinn. Sie dienen der Referenzierung von sogenannten BLOB-Objekten in Basistabellen. Die Definition von REF-Werten in Basistabellen wird im Abschnitt „Spalte“ auf Seite 92 beschrieben. Die Struktur und Bearbeitung der BLOB-Objekte ist im Abschnitt „BLOB-Konstrukte“ auf Seite 99 erklärt.

NULL-Werte

Um fehlende Werte von Werten zu unterscheiden, gibt es NULL-Werte. Der Begriff NULL-Wert stellt keinen bestimmten Wert dar, sondern bezeichnet den Sachverhalt „Wert unbekannt“ oder „irrelevant“. Insbesondere darf der NULL-Wert nicht als Leerzeichen oder als die Ziffer 0 interpretiert werden.

Das Schlüsselwort NULL kann als Literal für den NULL-Wert betrachtet werden. Es kann z.B. beim Einfügen (INSERT), Einfügen/Ändern (MERGE) und Ändern (UPDATE) angegeben werden, um für einen Spaltenwert den NULL-Wert einzutragen. Innerhalb einer Spaltendefinition kann auch mit der Klausel DEFAULT NULL der NULL-Wert als Defaultwert festgelegt werden.

Wird bei der Spaltendefinition NOT NULL oder PRIMARY KEY für eine Spalte angegeben, dann können nur Nicht-NULL-Werte in dieser Spalte enthalten sein. Ist keine NOT NULL- oder PRIMARY KEY-Bedingung und kein Defaultwert (siehe Seite 92) ungleich NULL definiert, dann trägt SESAM/SQL automatisch den NULL-Wert ein, wenn beim Einfügen eines Satzes für eine Spalte kein Wert angegeben wird.

Ein NULL-Wert in Bedingungen ergibt, abhängig von der Art der Verknüpfung (siehe Seite 125), in den meisten Fällen als Ergebnis den Wahrheitswert unbestimmt. Eine Ausnahme ist das Prädikat *spalte* IS [NOT] NULL, das nur die Wahrheitswerte wahr oder falsch ergibt.

Werte für multiple Spalten

Ein Wert für einen Bereich von Spaltenelementen einer multiplen Spalte wird mit einem *wert* angegeben, der eine Benutzervariable für einen Vektor, ein Platzhalter der Form „?“ oder ein Aggregat ist. Ein Aggregat wird in der folgenden Form angegeben:

<*wert1*,*wert2*, ...>

wert1, *wert2*, ... sind hierbei die Werte für die Ausprägungen der multiplen Spalte.

Zeichenketten

Zeichenketten sind eine Folge von beliebigen Zeichen in EBCDIC oder Unicode. EBCDIC-Zeichenketten werden als „alphanumerische Werte“, Unicode-Zeichenketten als „National-Werte“ bezeichnet.

In SESAM/SQL werden alphanumerische Literale, National-Literale und Spezial-Literale zur Darstellung von Zeichenketten verwendet.

Alphanumerische Werte

Alphanumerische Literale werden in der Form '[*zeichen*...]' bzw. X'[*hex hex*...]' angegeben und können jedes beliebige EBCDIC-Zeichen enthalten. Ein Hochkomma innerhalb eines alphanumerischen Literals muss verdoppelt werden; das verdoppelte Hochkomma gilt als ein Zeichen. Der Datentyp ist CHAR (*n*), wobei *n* die Anzahl der Zeichen ist.

Neben der Angabe von alphanumerischen Literalen gibt es die Möglichkeit, alphanumerische Werte über Spezial-Literale anzugeben. Z.B. liefert SYSTEM_USER den Namen des aktuellen Systembenutzers.

Für alphanumerische Werte gibt es den Operator ||. Er verbindet zwei Zeichenketten zu einer Zeichenkette (Konkatenation). Bei der Konkatenation von Zeichenketten müssen entweder beide Operanden alphanumerisch (CHAR oder VARCHAR) oder beide vom National-Typ (NCHAR oder NVARCHAR) sein.

Ein alphanumerisches Literal kann sich auch aus Teilketten zusammensetzen, die jeweils in einer Zeile stehen:

'*teilkette_1*'
'*teilkette_2*'
...
'*teilkette_n*'

Dies entspricht dem alphanumerischen Literal

'*teilkette_1*'||'*teilkette_2*'|| ... ||'*teilkette_n*'

Kommentare und Leerzeichen zwischen den Teilketten sind erlaubt.

National-Werte

National-Literale werden in der Form N'[*zeichen*...]', NX'[*4hex*...]' oder U&'[*zeichen ...esc+4hex...zeichen*...] angegeben.

N-Literale können Unicode-Zeichen enthalten, die auch im codierten Zeichensatz EDF03IRV enthalten sind. Ein Hochkomma innerhalb eines National-Literals muss verdoppelt werden; das verdoppelte Hochkomma gilt als ein Zeichen. Der Datentyp ist NCHAR (*cu_länge*), wobei *cu_länge* die Anzahl der Code Units ist (1 Code Unit in UTF-16 = 2 Byte).

NX-Literale enthalten nur Unicode-Zeichen in sedezimaler Darstellung.

U&-Literale enthalten sowohl Unicode-Zeichen aus dem codierten Zeichensatz EDF03IRV als auch Unicode-Zeichen in sedezimaler Darstellung, die mit einem vorangestellten Entwertungszeichen gekennzeichnet werden, U&'[*esc 4hex*...]' bzw. U&'[*esc+6hex*...]'. Das Literal kann beliebig viele sedezimale Zeichen an beliebigen Stellen enthalten, jedes einzelne Zeichen muss aber entwertet werden.

Für National-Werte gibt es den Operator ||, der zwei Zeichenketten zu einer Zeichenkette verbindet (Konkatenation). Bei der Konkatenation von Zeichenketten müssen entweder beide Operanden alphanumerisch (CHAR oder VARCHAR) oder beide vom National-Datentyp (NCHAR oder NVARCHAR) sein.

Ein National-Literal kann sich auch aus Teilketten zusammensetzen, die jeweils in einer Zeile stehen:

N'*teilkette_1*'
'*teilkette_2*'
...
'*teilkette_n*'

Dies entspricht dem National-Literal

N'*teilkette_1*'||N'*teilkette_2*'|| ... ||N'*teilkette_n*'

Spezial-Literale

Spezial-Literale werden zur Ablaufzeit in die entsprechenden Werte ihrer Ablaufumgebung umgewandelt:

Spezial-Literal	Bedeutung
CURRENT_CATALOG	Name der voreingestellten Datenbank
CURRENT_ISOLATION_LEVEL	Isolationslevel der aktuellen Transaktion
CURRENT_REFERENCED_CATALOG	Name der Datenbank, auf die sich die aktuelle Anweisung bezieht
CURRENT_SCHEMA	Name des voreingestellten Schemas
[CURRENT_]USER	Name des aktuellen Berechtigungsschlüssels
SYSTEM_USER	Name des aktuellen Systembenutzers

Tabelle 31: Spezial-Literale

Numerische Werte

Numerische Werte sind Ganzzahlen, Festpunktzahlen oder Gleitpunktzahlen.

Zeitwerte

SESAM/SQL unterscheidet die Zeitwerte

- Datum (`DATE'`*jahr-monat-tag*`'`)
- Uhrzeit (`TIME'`*stunde*`:`*minute*`:`*sekunde*`.`*sekundenbruchteil*`'`)
- Zeitstempel (`TIMESTAMP'`*jahr−monat−tag stunde*`:`*minute*`:`*sekunde*`.`*sekundenbruchteil*`'`).

Als Zeitwerte können auch die Zeitfunktionen (siehe Seite 121) CURRENT_DATE, CURRENT_TIME(3), LOCALTIME(3), CURRENT_TIMESTAMP(3) und LOCALTIMESTAMP(3) angegeben werden. Sie liefern das aktuelle Datum und/oder die aktuelle Uhrzeit.

Für *sekunde* kann eine Zahl zwischen 00 und 61 angegeben werden. Die Zahlen nach dem Punkt geben die dreistelligen Sekundenbruchteile an.

Zeitwerte können in Spalten vom entsprechenden Datentyp eingetragen werden.
Zeitwerte werden verwendet

- in den Zeitfunktionen CURRENT_DATE, CURRENT_TIME(3), LOCALTIME(3), CURRENT_TIMESTAMP(3) und LOCALTIMESTAMP(3)
- in den Mengenfunktionen COUNT, MAX und MIN
- in den numerischen Funktionen EXTRACT und JULIAN_DAY_OF_DATE
- in Vergleichen mit einem anderen Zeitwert des gleichen Datentyps.

4.3.4 Ausdrücke

Die Auswertung eines Ausdrucks ergibt einen Wert, also einen alphanumerischer Wert, einen National-Wert, einen numerischer Wert oder einen Zeitwert.

Der Datentyp dieses Werts bestimmt, ob es sich bei dem Ausdruck um einen alphanumerischen Ausdruck, einen National-Ausdruck, einen numerischen Ausdruck oder einen Zeitwerte-Ausdruck handelt.

Ausdrücke können vorkommen in:

- Spaltenauswahl (SELECT-Ausdruck, SELECT-Anweisung)
- Prädikaten von Suchbedingungen (z.B. WHERE-, HAVING-Klausel, IF-Anweisung)
- Zuweisungen (INSERT-, MERGE, UPDATE- oder SET-Anweisung)

Ein Ausdruck besteht aus den folgenden Operanden und eventuell Operatoren.
Die Operatoren werden auf die Ergebnisse der Operanden angewendet.

- Werte (siehe Seite 116)
  - als Zeichenketten (alphanumerische Literale und National-Literale)
  - als Spezial-Literale
  - als Benutzervariablen
  - als Parameter
  - als lokale Variable
  - als Platzhalter in dynamisch übersetzbaren SQL-Anweisungen
- Spalten (siehe Seite 92)
- Funktionen (siehe Seite 121)
  - Zeitfunktionen
  - numerische Funktionen
  - Zeichenkettenfunktionen
  - Mengenfunktionen
  - Tabellenfunktionen
  - kryptografische Funktionen
  - User Defined Functions (UDF)
- CASE-Ausdruck (siehe Seite 124)
- CAST-Ausdruck (siehe Seite 124)

- Unterabfragen (siehe Seite 130)

4.3.5 Funktionen

Eine Funktion besteht aus ihrem Funktionsnamen und Parametern. Sie berechnet einen Wert oder liefert eine Tabelle (Tabellenfunktion). Funktionen können innerhalb von Ausdrücken vorkommen. Bei der Auswertung eines Ausdrucks wird eine darin vorkommende Funktion ausgeführt und durch den berechneten Wert oder die gelieferte Tabelle ersetzt.

Die SESAM/SQL-Funktionen sind in folgende Gruppen eingeteilt:

- Zeitfunktionen
- Zeichenkettenfunktionen
- Numerische Funktionen
- Mengenfunktionen
- Tabellenfunktionen
- kryptografische Funktionen
- User Defined Functions (UDF)

4.3.5.1 Zeitfunktionen

Zeitfunktionen ermitteln

- das aktuelle Datum (CURRENT_DATE)
- die aktuelle Uhrzeit (CURRENT_TIME(3) oder LOCALTIME(3))
- einen Zeitstempel mit aktuellem Datum und aktueller Uhrzeit (CURRENT_TIMESTAMP(3) oder LOCALTIMESTAMP(3))
- die einem ganzzahligen Wert entsprechende julianische Tagesnummer (DATE_OF_JULIAN_DAY)
  (siehe auch die inverse Funktion JULIAN_DAY_OF_DATE auf Seite 122).

LOCALTIMESTAMP(3) und CURRENT_TIMESTAMP(3) bzw. LOCALTIME(3) und CURRENT_TIME(3) sind in SESAM/SQL äquivalent.
In der SQL-Norm sind CURRENT_TIMESTAMP und CURRENT_TIME für eine Spracherweiterung vorgesehen, bei der mit unterschiedlichen Zeitzonen gerechnet wird.

4.3.5.2 Zeichenkettenfunktionen

Zeichenkettenfunktionen

- extrahieren Teilzeichenketten (SUBSTRING)
- transliterieren alphanumerische Zeichenketten in National-Zeichenketten und umgekehrt (TRANSLATE)
- transcodieren National-Zeichenketten von UTFE nach UTF-16 und umgekehrt (TRANSLATE)
- entfernen führende oder nachgestellte Zeichen von Zeichenketten (TRIM)
- wandeln Groß- in Kleinbuchstaben bzw. Klein- in Großbuchstaben um (LOWER, UPPER)
- konvertieren einen Wert von beliebigem Datentyp in die interne Darstellung (als alphanumerische Zeichenkette oder hexadezimal) und umgekehrt (HEX_OF_VALUE, VALUE_OF_HEX, REP_OF_VALUE, VALUE_OF_REP)
- liefern für National-Zeichenketten das Collation-Element gemäß der Default Unicode Translation Table (COLLATE)
- bringen National-Zeichenketten in Normalform (NORMALIZE)

4.3.5.3 Numerische Funktionen

Numerische Funktionen erfüllen unterschiedliche Zwecke:

- ABS(), CEILING(), FLOOR(), MOD(), SIGN() und TRUNC() führen die entsprechenden mathematischen Funktionen auf den angegebenen numerischen Ausdrücken aus.
- CHARACTER_LENGTH( ), OCTET_LENGTH( ) und POSITION( ) berechnen die Anzahl der Bytes oder Code Units einer Zeichenkette bzw. die Position einer Zeichenkette innerhalb einer anderen Zeichenkette.
- JULIAN_DAY_OF_DATE( ) wandelt ein Datum in einen ganzzahligen Wert um.
- EXTRACT() extrahiert bestimmte Bestandteile eines Zeitwertes (z.B. den Tag im Jahr) .

Bei Auswertung einer numerischen Funktion wird ein numerischer Wert zurückgeliefert.

4.3.5.4 Mengenfunktionen

Mengenfunktionen bestimmen Durchschnitt (AVG), Anzahl (COUNT), Maximum (MAX), Minimum (MIN) und Summe (SUM) einer Menge von Werten bzw. die Anzahl der Sätze einer Ergebnistabelle.

Die Mengenfunktionen MIN() und MAX() beziehen sich auf die Menge aller Werte einer Spalte in einer Tabelle. Sie unterscheiden sich dadurch von einem CASE-Ausdruck mit MIN / MAX (siehe Seite 124), der sich auf unterschiedliche Ausdrücke bezieht.

4.3.5.5 Tabellenfunktionen

Tabellenfunktionen erzeugen Tabellen, deren Inhalt von den Aufrufparametern abhängt oder aus externe Datenquellen, z.B. Dateien, stammt.

In SESAM/SQL gibt es folgende Tabellenfunktionen

- CSV()
  liefert eine Tabelle, deren Werte aus einer BS2000-Datei (der so genannten CSV-Datei) gelesen werden
- DEE()
  liefert eine Tabelle mit einer Zeile ohne Spalte

Für die Darstellung von SQL-Tabellen in Dateien wird dabei das CSV-Format verwendet (CSV: Comma Separated Values). Das ist ein standardisiertes Format für den plattform-unabhängigen Austausch von tabellarischen Daten. Die Datei enthält die Folge von Zeilen der Tabelle, wobei jede Zeile der Reihe nach ihre Spaltenwerte als Zeichenketten enthält. Solche Dateien können mit vielen Softwareprodukten (z.B. mit Microsoft EXCEL) erzeugt werden. Nähere Informationen zum CSV-Format und zur Interpretation von CSV-Dateien in SESAM/SQL finden Sie im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“.

4.3.5.6 Kryptografische Funktionen

Die Funktionen ENCRYPT() und DECRYPT() dienen der Verschlüsselung und Entschlüsselung von einzelnen Werten. Sensitive Daten werden durch die Verschlüsselung vor unbefugtem Zugriff geschützt. Nur die Benutzer, die den „Schlüssel“ („key“) kennen, können die Daten entschlüsseln.

Einführende Informationen zum Zugriffsschutz durch Datenverschlüsselung in SESAM/SQL finden Sie im Abschnitt „Zugriffsschutz durch Datenverschlüsselung“ auf Seite 193.

4.3.5.7 User Defined Functions (UDF)

In einer UDF werden Abläufe von SQL-Anweisungen in der Datenbank gespeichert, die später mit einem einzigen Funktionsaufruf ausgeführt werden können. UDFs haben einen fast identischen Funktionsumfang wie Prozeduren, siehe Abschnitt „Routine“ auf Seite 98.

4.3.6 CASE-Ausdruck

Ein CASE-Ausdruck ist ein bedingter Ausdruck, d.h. ein Ausdruck, der Bedingungen beinhaltet. Jeder Bedingung ist ein Ausdruck bzw. der NULL-Wert zugeordnet.
Bei der Auswertung des CASE-Ausdrucks wird der zugeordnete Ausdrucks- bzw. NULL-Wert derjenigen Bedingung zurückgeliefert, die wahr ist.

Es gibt verschiedene Varianten des CASE-Ausdrucks:

- CASE-Ausdruck mit Suchbedingung
- Einfacher CASE-Ausdruck
- CASE-Ausdruck mit NULLIF
- CASE-Ausdruck mit COALESCE
- CASE-Ausdruck mit MIN oder MAX

Ein CASE-Ausdruck mit MIN / MAX bezieht sich auf unterschiedliche Ausdrücke. Er unterscheidet sich dadurch von den Mengenfunktionen MIN() und MAX() (siehe Seite 123), die sich auf die Menge aller Werte einer Spalte in einer Tabelle beziehen.

Mit Hilfe von CASE-Ausdrücken können z.B. Spaltenwerte umcodiert oder bestimmte Spaltenwerte durch NULL-Werte ersetzt werden.

4.3.7 CAST-Ausdruck

Mit Hilfe des CAST-Ausdrucks kann ein Wert eines bestimmten Datentyps in einen Wert eines anderen Datentyps umgewandelt werden.

4.3.8 Suchbedingung

Eine Suchbedingung ist eine Bedingung, die zur Auswahl von Sätzen dient. Es werden nur die Sätze ausgewählt, für die die angegebene Suchbedingung den Wahrheitswert „wahr“ liefert. Suchbedingungen können in mehreren SQL-Anweisungen auftreten, die eine Auswahl von Sätzen in einer WHERE-, HAVING- oder ON-Klausel ermöglichen. Außerdem können Suchbedingungen in der CHECK-Bedingung (siehe Seite 95) und in einem CASE-Ausdruck (siehe Seite 124) auftreten.

Eine Suchbedingung besteht aus Prädikaten (logischen Ausdrücken), die durch die logischen Operatoren AND, OR und NOT verknüpft sein können.

Eine Suchbedingung wird ausgewertet, indem die Prädikate ausgewertet und die logischen Operatoren auf die Ergebnisse angewendet werden. Für die Auswertung gelten die üblichen Vorrangregeln: NOT vor AND vor OR. Durch Klammern kann die Reihenfolge der Auswertung beeinflusst werden.

Das Ergebnis einer Suchbedingung ist der Wahrheitswert wahr, falsch oder unbestimmt.

AND **Operand2** **Operand1**	**wahr**	**falsch**	**unbestimmt**
wahr	wahr	falsch	unbestimmt
falsch	falsch	falsch	falsch
unbestimmt	unbestimmt	falsch	unbestimmt

Tabelle 32: Wahrheitswerte bei logischer UND-Verknüpfung

OR **Operand2** **Operand1**	**wahr**	**falsch**	**unbestimmt**
wahr	wahr	wahr	wahr
falsch	wahr	falsch	unbestimmt
unbestimmt	wahr	unbestimmt	unbestimmt

Tabelle 33: Wahrheitswerte bei logischer ODER-Verknüpfung

NOT **Operand1**	
wahr	falsch
falsch	wahr
unbestimmt	unbestimmt

Tabelle 34: Wahrheitswerte bei logischer Negation

4.3.9 Prädikate

Prädikate sind die Bestandteile von Suchbedingungen.
Ein Prädikat besteht aus Operanden und Operatoren.

Entsprechend den Operatoren sind Prädikate in folgende Gruppen unterteilt:

- Vergleich von zwei Zeilen
  Zwei Zeilen werden mit einem Vergleichsoperator lexikografisch verglichen. Haben beide Zeilen nur eine Spalte, dann erhält man den gewöhnlichen Vergleich von zwei Werten. Als Vergleichsoperatoren sind zugelassen:

=	gleich
<	kleiner
<=	kleiner oder gleich
>	größer
>=	größer oder gleich
<>	ungleich

- Quantifizierter Vergleich (Vergleich mit den Zeilen einer Tabelle)
  Der Wert einer Zeile wird mit den Zeilen einer Tabelle verglichen. Entweder wird bestimmt, ob der Vergleich für alle Zeilen der Tabelle zutrifft (ALL), oder ob er wenigstens für eine Zeile zutrifft (SOME/ANY).
- BETWEEN-Prädikat (Bereichsabfrage)
  Es wird bestimmt, ob eine Zeile in einem Bereich (BETWEEN) liegt oder nicht (NOT BETWEEN), der durch eine obere und untere Grenze angegeben wird.
- CASTABLE-Prädikat (Konvertierbarkeit prüfen)
  Es wird geprüft, ob ein Ausdruck in einen bestimmten Datentyp konvertiert werden kann. Mit dem CASTABLE-Prädikat können Sie die Ausführbarkeit eines entsprechenden CAST-Ausdruckes (siehe Abschnitt „CAST-Ausdruck“ auf Seite 124) schon vor seiner Ausführung prüfen und geeignet reagieren.
- IN-Prädikat (Elementabfrage)
  Es wird bestimmt, ob eine Zeile in einer Tabelle vorkommt (IN) oder nicht (NOT IN).
- LIKE-Prädikat (einfacher Mustervergleich)
  Es wird geprüft, ob ein alphanumerischer Wert oder ein National-Wert zu einem angegebenen Muster passt.

- LIKE_REGEX-Prädikat (Mustervergleich mit regulären Ausdrücken)
  Es wird geprüft, ob ein alphanumerischer Wert zu einem angegebenen regulären Ausdruck passt. Reguläre Ausdrücke sind genau definierte Suchmuster, die weit über die Möglichkeiten der Suchmuster im LIKE-Prädikat hinaus gehen. Reguläre Ausdrücke sind ein mächtiges Mittel, um große Datenbestände nach komplexen Suchausdrücken zu durchsuchen. Sie werden seit langem z.B. in der Programmiersprache Perl eingesetzt.
- NULL-Prädikat (Vergleich auf NULL-Wert)
  Es wird geprüft, ob eine Spalte, ein Parameter oder eine lokale Variable den NULL-Wert enthält.
- EXISTS-Prädikat (Existenzabfrage)
  Es wird geprüft, ob die angegebene Ergebnistabelle nicht leer ist.

Für Vergleiche müssen die Datentypen der Operanden vergleichbar sein.
Alle numerischen Werte sind mit numerischen Werten, alle alphanumerischen Zeichenketten sind mit alphanumerischen Zeichenketten, alle National-Zeichenketten sind mit National-Zeichenketten vergleichbar. Datum, Zeit und Zeitstempel sind jeweils nur mit Datum, Zeit bzw. Zeitstempel vergleichbar.

4.3.10 Abfrage-Ausdruck

Ein Abfrage-Ausdruck ist das Sprachmittel in SQL, um Sätze und Spalten aus Basistabellen und Views auszuwählen. Das Ergebnis eines Abfrage-Ausdrucks ist eine Ergebnistabelle, die die ausgewählten Sätze enthält.

Abfrage-Ausdrücke werden z.B. verwendet, um einen View oder Cursor zu definieren oder um eine Menge von Sätzen auszuwählen, die mit INSERT in eine Tabelle eingefügt werden.

Der UNION-Operator verbindet zwei Abfrageausdrücke derart, dass die entstehende Ergebnistabelle alle Sätze enthält, die in mindestens einer der Ergebnistabellen der durch UNION verknüpften Abfrage-Ausdrücke vorkommen.

Die EXCEPT-Operation im Abfrageausdruck ist vergleichbar mit der Differenz zweier Mengen in der Mengenlehre.

Änderbarer Abfrage-Ausdruck

Für die Definition eines änderbaren View oder eines änderbaren Cursors spielt der Begriff des änderbaren Abfrage-Ausdrucks eine Rolle. Ein Abfrage-Ausdruck ist änderbar, wenn er folgende Bedingungen erfüllt:

- Der Abfrage-Ausdruck enthält keinen Join-Ausdruck.
  Join-Ausdrücke werden im Abschnitt „Join-Ausdruck“ auf Seite 131 dargestellt.
- Der Abfrage-Ausdruck enthält keine UNION-Operation und keine EXCEPT-Operation.
- In der SELECT-Liste dürfen nur Spaltennamen angegeben werden. Andere Elemente eines Ausdrucks, beispielsweise Unterabfragen, Funktionsaufrufe oder Literale, sind nicht erlaubt. Einfache Spalten dürfen nicht mehrfach angegeben werden. Teilbereiche von multiplen Spalten dürfen sich nicht überlappen.
- In der FROM-Klausel darf nur eine Tabelle angegeben sein oder eine änderbare Unterabfrage. Ist eine Tabelle angegeben, muss sie eine Basistabelle oder ein änderbarer View sein.
- In der WHERE-Klausel darf keine Unterabfrage vorkommen.
- Das Schlüsselwort DISTINCT darf nicht angegeben werden.
- Der SELECT-Ausdruck darf keine GROUP BY- oder HAVING-Klausel enthalten.

4.3.10.1 SELECT-Ausdruck

Der SELECT-Ausdruck ist ein Abfrage-Ausdruck, der durch das Schlüsselwort SELECT eingeleitet wird. Das Ergebnis eines SELECT-Ausdrucks ist eine Ergebnistabelle.

Der SELECT-Ausdruck setzt sich aus einzelnen Klauseln zusammen, mit denen jeweils bestimmte Operationen verbunden sind. Diese Klauseln führen insgesamt zu der endgültigen Ergebnistabelle.

- SELECT-Liste: hier werden die Spalten der Ergebnistabelle festgelegt.
- FROM-Klausel: hier werden die Tabellen angegeben; ihre Ergebnistabelle ist deren Kartesisches Produkt.

  Werden zwei oder mehrere Tabellen in der FROM-Klausel angegeben, wird das sogenannte Kartesische Produkt dieser Tabellen gebildet. Das Kartesische Produkt zweier Tabellen wird gebildet, indem jeder Satz der ersten Tabelle mit jedem Satz der zweiten Tabelle verkettet wird. Die Anzahl der Sätze des Kartesischen Produkts entspricht somit dem Produkt der Anzahl der Sätze der zu Grunde liegenden Tabellen.
- WHERE-Klausel: hier wird eine Suchbedingung angegeben. Ihre Ergebnistabelle besteht aus den Sätzen der Ergebnistabelle der FROM-Klausel, die die Suchbedingung erfüllen.

  Eine besondere Form der Suchbedingung in der WHERE-Klausel ist die **Join-Bedingung**. Damit können Daten aus zwei oder mehr Tabellen verknüpft werden. Eine Join-Bedingung über zwei Tabellen wird formuliert, indem eine Spalte der einen Tabelle mit einer entsprechenden Spalte der anderen Tabelle durch einen Vergleichsoperator verbunden wird. Aus dem Kartesischen Produkt der beiden Tabellen werden nur Sätze ausgewählt, die dieser Join-Bedingung genügen. Eine solche Verknüpfung von Tabellen (wie auch manchmal die entstehende Ergebnistabelle) nennt man einen Join.

  Neben dieser Möglichkeit, Tabellen durch eine Join-Bedingung in der WHERE-Klausel zu verbinden, gibt es die umfassendere Möglichkeit, Joins durch Join-Ausdrücke zu formulieren. Join-Ausdrücke werden auf Seite 131 beschrieben.
- GROUP BY-Klausel: hier können Gruppierungsmerkmale angegeben werden. Ihre Ergebnistabelle ist die der WHERE-Klausel, gruppiert nach Gruppen mit gleichen Werten der Gruppierungsmerkmale.
  Fehlt die GROUP BY-Klausel, so wird nicht gruppiert.
- HAVING-Klausel: hier kann eine Suchbedingung angegeben werden. Die Ergebnistabelle besteht aus den Gruppen, die die Suchbedingung erfüllen.

Die Reihenfolge der Klauseln im SELECT-Ausdruck muss eingehalten werden. Die schrittweise Bildung der Ergebnistabelle folgt derselben Reihenfolge, wie sie im SELECT-Ausdruck vorgegeben ist. Eine Ausnahme ist die SELECT-Liste, die erst nach der Auswertung der FROM-, WHERE-, GROUP BY- und HAVING-Klausel ausgewertet wird.

4.3.10.2 Unterabfrage

Eine Unterabfrage ist ein Abfrage-Ausdruck, der in folgenden Fällen verwendet werden kann:

- In Ausdrücken:
  Die Unterabfrage muss eine einspaltige Ergebnistabelle mit höchstens einem Satz liefern. Der Wert der Unterabfrage ist dann der Wert in der Ergebnistabelle bzw. der NULL-Wert, wenn die Ergebnistabelle leer ist.
- In Prädikaten:
  In den Prädikaten ANY, SOME, ALL, IN und EXISTS liefert die Unterabfrage eine Ergebnistabelle.
- In der FROM-Klausel von SELECT-Ausdrücken:
  Die Unterabfrage liefert eine Ergebnistabelle.
- In Join-Ausdrücken:
  Die Unterabfrage liefert eine Ergebnistabelle.

Eine Unterabfrage wird immer in runde Klammern eingeschlossen.

Korrelierte Unterabfrage

Bei einem geschachtelten Abfrage-Ausdruck heißt eine innere Unterabfrage korrelierte Unterabfrage, wenn sie sich auf Spalten einer Tabelle bezieht, die in einem äußeren Abfrage-Ausdruck verwendet wird.

Korrelierte Unterabfragen werden ausgewertet, indem die innere Unterabfrage für jeden Satz der Tabelle in der äußeren Abfrage ausgewertet wird. Nicht korrelierte Unterabfragen werden nur einmal ausgewertet, da sie nicht von der äußeren Abfrage abhängen.

4.3.10.3 Join-Ausdruck

Bei der Beschreibung der WHERE-Klausel auf Seite 128 wurde die einfachste Form des Joins vorgestellt, die es ermöglicht, Tabellen mit Hilfe einer Join-Bedingung zu verbinden. Eine wesentliche Erweiterung der Funktionalität bei der Verwendung eines Joins bietet der Join-Ausdruck.

Ein Join-Ausdruck enthält die zu verknüpfenden Tabellen, die gewünschte Join-Operation und gegebenenfalls eine Join-Bedingung.

Ein Join-Ausdruck kann angegeben werden:

- als Abfrage-Ausdruck in einer SQL-Anweisung
- in der FROM-Klausel eines SELECT-Ausdrucks oder einer SELECT-Anweisung
- in einer Unterabfrage in Select-Liste und HAVING-Klausel

Die Ergebnistabelle eines Join-Ausdrucks ist nicht änderbar.

Werden zwei Tabellen mit einem Join verbunden, spricht man von einem einfachen Join. Werden mehr als zwei Tabellen verbunden, spricht man von einem zusammengesetzten Join. Um einen zusammengesetzten Join zu bilden, können Join-Ausdrücke wiederum in Join-Ausdrücke eingesetzt werden. Bei solchen geschachtelten Joins kann durch Angabe von Klammern die Gruppierung der Operatoren beeinflusst werden.

Die Schlüsselwörter CROSS, INNER, OUTER und UNION legen den Typ des Joins fest.

CROSS JOIN

Ein CROSS JOIN zwischen zwei Tabellen ergibt das Kartesische Produkt (Kreuzprodukt) dieser Tabellen. In der Ergebnistabelle eines CROSS JOIN wird jeder Satz der ersten Tabelle mit jedem Satz der zweiten Tabelle verkettet.

INNER JOIN

INNER legt einen sogenannten inneren Join (INNER JOIN) fest. Bei einem inneren Join werden aus dem Kartesischen Produkt nur die Sätze in die Ergebnistabelle übernommen, deren Join-Spalten die Join-Bedingung erfüllen.

OUTER JOIN

OUTER legt einen sogenannten äußeren Join (OUTER JOIN) fest. Ausgangstabelle für einen äußeren Join ist wie beim inneren Join das Kartesische Produkt. Beim einseitigen äußeren Join unterscheidet man eine dominante und eine abhängige Tabelle. Im Gegensatz zum inneren Join enthält der äußere Join auch dann einen Satz, wenn für einen Satz der dominanten Tabelle kein Satz der abhängigen Tabelle vorhanden ist, mit dem die Join-Bedingung erfüllt ist. Für den nicht vorhandenen Satz der abhängigen Tabelle wird ein Satz mit NULL-Werten verwendet.
Das Schlüsselwort LEFT OUTER legt die Tabelle links vom LEFT OUTER-Operator, das Schlüsselwort RIGHT OUTER die Tabelle rechts vom RIGHT OUTER-Operator als dominante Tabelle fest. Bei einem FULL OUTER -Join werden die Tabellen links und rechts vom FULL-Operator je einmal als dominante und als abhängige Tabelle festgelegt. Die Ergebnistabelle des FULL OUTER-Join enthält alle Sätze des LEFT OUTER- und des RIGHT OUTER-Join. Jeder Satz, der sowohl in der Ergebnistabelle des LEFT OUTER- als auch des RIGHT OUTER-Join enthalten ist, wird nur einmal in die Ergebnistabelle des FULL OUTER-Join übernommen. Duplikate innerhalb der LEFT OUTER-Tabelle bzw. innerhalb der RIGHT OUTER-Tabelle werden als Duplikate in die Ergebnistabelle des FULL OUTER-Join übernommen.
Innere und äußere Joins können in einem Join-Ausdruck kombiniert werden.

UNION JOIN

Anders als beim inneren und beim äußeren Join ist beim UNION JOIN nicht das Kartesische Produkt die Ausgangstabelle. Stattdessen werden die Tabellen links und rechts vom UNION-Operator als Ausgangstabellen verwendet. Der UNION JOIN wird wie folgt gebildet:

- Die erste Tabelle des UNION JOIN wird nach rechts um die Spalten der zweiten Tabelle erweitert; für die Spaltenwerte der zweiten Tabelle werden NULL-Werte eingetragen.
- Die zweite Tabelle wird nach links um die Spalten der ersten Tabelle erweitert; für die Spaltenwerte der ersten Tabelle werden NULL-Werte eingetragen.
- Die um NULL-Werte erweiterte erste Tabelle wird mit UNION mit der um NULL-Werte erweiterten zweiten Tabelle vereinigt.

4.3.11 SQL-Datenmanipulation ohne Cursor

SQL-Anweisungen zum Abfragen und Ändern von Daten lassen sich wie folgt unterscheiden:

- SQL-Anweisungen, die über einen Cursor auf Sätze zugreifen
  Die Funktion eines Cursors und die SQL-Anweisungen, mit denen ein Cursor angesprochen wird, werden im Abschnitt „SQL-Datenmanipulation mit Cursor“ auf Seite 141 dargestellt.
- SQL-Anweisungen, die ohne Verwendung eines Cursors auf Sätze zugreifen
  Folgende SQL-Anweisungen verwenden keinen Cursor:

SELECT	wählt Werte eines Satzes aus und überträgt diese Werte in Benutzervariablen (siehe Seite 139)
INSERT	fügt Sätze in eine bestehende Tabelle ein
MERGE	vereint die Funktionen INSERT und UPDATE in einem Arbeitsgang. Abhängig vom Ergebnis der ON-Bedingung ändert MERGE Spaltenwerte von bereits existierenden Sätzen oder fügt neue Sätze in eine bestehende Tabelle ein.
UPDATE	ändert Spaltenwerte in ausgewählten Sätzen einer Tabelle
DELETE	löscht ausgewählte Sätze einer Tabelle

Um Änderungen in einer Tabelle mit den obigen Anweisungen durchzuführen, muss

- der Anwender das entsprechende Privileg besitzen (siehe Seite 176), bei Änderungen mit Hilfe eines Abfrage-Ausdrucks oder einer Suchbedingung zusätzlich das SELECT-Privileg
- der Transaktionsmodus (siehe Seite 134) der aktuellen Transaktion READ WRITE sein
- in der DBH-Startanweisung ADD-SQL-DATABASE-CATALOG-LIST (siehe Handbuch „Datenbankbetrieb“) entweder ACCESS=*PARAMETERS (WRITE=*YES) oder ACCESS=*PARAMETERS (CAT-ADMINISTRATION=*YES) eingestellt sein.

4.4 SQL-Transaktion

Eine SQL-Transaktion ist eine Folge von zusammengehörigen SQL-Anweisungen, die eine Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt. Änderungen in Tabellen sind nach Ende einer SQL-Transaktion entweder vollständig oder überhaupt nicht ausgeführt.

Eine SQL-Transaktion beginnt, wenn keine andere Transaktion im Gang ist und eine SQL-Anweisung ausgeführt wird, die eine Transaktion einleitet. SQL-Anweisungen, die eine Transaktion einleiten, sind alle SQL-Anweisungen außer den folgenden SQL-Anweisungen:

- ALTER TABLE mit dem Pragma UTILITY MODE ON
- DECLARE CURSOR (nicht ausführbar)
- PERMIT
- SET CATALOG
- SET SCHEMA
- SET SESSION AUTHORIZATION
- SET TRANSACTION
- WHENEVER (nicht ausführbar)
- Utility-Anweisungen

Die Anweisungen EXECUTE und EXECUTE IMMEDIATE leiten eine Transaktion ein, wenn die betreffende dynamisch ausführbare Anweisung eine Transaktion einleitet.

In SESAM/SQL dürfen SQL-Anweisungen zum Abfragen und Ändern von Daten (siehe Seite 107) nicht in einer Transaktion ausgeführt werden, in der eine SQL-Anweisung zur Schemadefinition und -verwaltung, zur Verwaltung der Speicherstruktur oder zur Verwaltung von Benutzereinträgen ausgeführt wird. Die Anweisungen SET SESSION AUTHORIZATION und SET TRANSACTION sowie Utility-Anweisungen können nur außerhalb einer Transaktion ausgeführt werden. Eine Utility-Anweisung wird über eine Folge interner Transaktionen abgewickelt und ist deshalb nicht rücksetzbar.

Eine SQL-Transaktion endet, wenn die SQL-Anweisung COMMIT [WORK] oder die SQL-Anweisung ROLLBACK [WORK] ausgeführt wird, bzw., wenn sie intern vom DBH zurückgesetzt wird.

Änderungen in der Datenbank seit dem Beginn einer bestimmten Transaktion werden erst festgeschrieben, wenn eine COMMIT-Anweisung erfolgreich durchgeführt wurde (*Ausnahme*: autonome Transaktionen, siehe Seite 137).

Wird eine Transaktion durch ROLLBACK beendet, dann werden alle Änderungen in der Datenbank seit dem Beginn der Transaktion rückgängig gemacht.

Tritt innerhalb einer Transaktion ein nicht behebbarer Fehler, ein Longlock oder Deadlock auf, führt SESAM/SQL einen impliziten ROLLBACK WORK aus.

Wird mit openUTM gearbeitet, dann wird die Transaktionsbeendigung ausschließlich mit UTM-Sprachmitteln durchgeführt. In UTM-Anwendungen dürfen die SQL-Anweisungen COMMIT und ROLLBACK nicht verwendet werden. Eine UTM-Transaktion endet mit dem Setzen eines Sicherungspunktes.

Eine SQL-Transaktion ist durch einen bestimmten Isolationslevel (siehe Seite 136) und einen bestimmten Transaktionsmodus (siehe Seite 137) charakterisiert. Isolationslevel und Transaktionsmodus können mit der SQL-Anweisung SET TRANSACTION festgelegt werden. Die mit SET TRANSACTION festgelegten Einstellungen sind jeweils nur für die SQL-Anweisungen der unmittelbar folgenden Transaktion gültig. Dabei ist es gleichgültig, ob es sich um eine SQL- oder CALL-DML-Transaktion oder eine gemischte Transaktion handelt. Nach dem Ende der Transaktion gelten wieder die Voreinstellungen.

Mit einer autonomen Transaktion (siehe Seite 137) können Sie Daten unabhängig vom Ausgang der umgebenden Transaktion in eine Datenbank schreiben.

Isolationslevel

Der Isolationslevel gibt an, wie stark das konsistente Lesen von Sätzen in einer Transaktion durch schreibende konkurrierende Zugriffe einer anderen Transaktion beeinträchtigt werden darf. Konkurrierende Zugriffe sind gleichzeitige Zugriffsversuche von zwei oder mehreren Anwenderprogrammen auf einen Satz. Folgende Phänomene können bei konkurrierenden Zugriffen je nach gewähltem Isolationslevel auftreten:

- dirty read (schmutziges Lesen)
  Eine Transaktion ändert einen Satz oder nimmt einen Satz neu auf. Eine zweite Transaktion liest diesen geänderten bzw. neuen Satz, bevor die erste Transaktion die Änderung festgeschrieben hat. Damit hat die zweite Transaktion einen Satz gelesen, der von der ersten noch geändert oder gelöscht werden kann, also noch nicht den endgültigen Zustand hat.
- non-repeatable read (nicht wiederholbares Lesen)
  Eine Transaktion liest einen Satz. Während diese Transaktion noch offen ist, ändert oder löscht eine zweite Transaktion diesen Satz und schreibt diese Änderung oder Löschung fest. Ein erneuter Zugriffsversuch der ersten Transaktion auf diesen Satz liefert dann veränderte Werte oder bleibt erfolglos.
- phantoms (Phantome)
  Eine Transaktion wählt mit einer Abfrage Sätze aus einer Tabelle auf Grund einer bestimmten Bedingung aus. Während diese Transaktion noch offen ist, nimmt eine zweite Transaktion Sätze in die Tabelle auf, die ebenfalls dieser Bedingung genügen. Wiederholt die erste Transaktion dieselbe Abfrage, so enthält die Ergebnistabelle auch die neu aufgenommenen Sätze.

Aus Kompatibilitätsgründen kann statt des Isolationslevels auch mit CONSISTENCY LEVEL der Konsistenzlevel angegeben werden.

Die folgende Tabelle zeigt die Zuordnung von Konsistenzlevel, Isolationslevel und Phänomenen, die jeweils auftreten können:

Isolationslevel	**Konsistenzlevel**	**dirty read**	**non-repeatable read**	**phantoms**
READ UNCOMMITTED	0	x	x	x
-	1	x	x[1]	x
READ COMMITTED	2	-	x	x
REPEATABLE READ	3	-	-	x
SERIALIZABLE	4	-	-	-

Tabelle 35: Isolationslevel, Konsistenzlevel und zugeordnete Phänomene

[1] Das Phänomen non-repeatable read kann auftreten, wenn vorher ein Satz mit dirty read gelesen wurde.

Voreinstellung für den Isolationslevel ist SERIALIZABLE, bei dem ein vollständiger Schutz vor konkurrierenden Transaktionen gewährleistet ist. Falls Isolationslevel bzw. Konsistenzlevel in der Konfigurationsdatei des Anwenderprogramms eingetragen sind, gilt der dort eingetragene Wert als Voreinstellung. Dieser Wert darf nicht höher sein als der, der durch die DBH-Option MAX-ISOLATION-LEVEL zugelassen ist. Anderenfalls meldet SESAM/SQL den SQLSTATE 91SCL.

Transaktionsmodus

Der Transaktionsmodus legt fest, ob innerhalb einer Transaktion Sätze nur gelesen oder auch geändert werden dürfen. READ ONLY ist Voreinstellung beim Isolationslevel READ UNCOMMITTED bzw. bei den Konsistenzleveln 0 und 1. Sonst ist READ WRITE die Voreinstellung. READ UNCOMMITTED ist nicht zulässig, wenn gleichzeitig als Transaktionsmodus READ WRITE festgelegt wurde.

Autonome Transaktion

Das Pragma AUTONOMOUS TRANSACTION ermöglicht es, Daten unabhängig vom Ausgang der umgebenden Transaktion in eine Datenbank zu schreiben. Insbesondere werden die Daten persistent in die Datenbank geschrieben, bevor möglicherweise die SQL-Anweisung ROLLBACK WORK der Transaktion ausgeführt wird.

Das Pragma wirkt nur bei ändernden SQL-Anweisungen, also bei INSERT, UPDATE, DELETE (mit Suchbedingung), MERGE und CALL.

Die SQL-Anweisung hinter dem Pragma AUTONOMOUS TRANSACTION wird in der aktuellen Transaktion des Anwenders, aber in einer eigenen Ablaufumgebung (eigenen Thread, eigener Transaktionskontext) ausgeführt.

SQL-Anweisungen des Anwenders zur Transaktionsverwaltung haben keine Wirkung. D.h. die Anweisungen COMMIT WORK bzw. ROLLBACK WORK des Anwenders beeinflussen das persistente Schreiben von Daten durch ändernde SQL-Anweisungen autonomer Transaktionen nicht. Die Anweisung SET TRANSACTION des Anwenders wirkt sich nicht auf autonome Transaktionen aus. Der Transaktionsmodus autonomer Transaktionen ist READ/WRITE und der Isolationslevel ist der maximale Wert, den die DBH-Option erlaubt.

Hinweise zur Benutzer-Identifikation, zu Lock-Konflikten und zum Abbruch der Anwendung finden Sie bei der Beschreibung des Pragmas AUTONOMOUS TRANSACTION im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“.

4.5 Umstellen von CALL-DML-Anwendungen

Um die Umstellung von CALL-DML-Anwendungen auf die SQL-Schnittstelle zu erleichtern, können innerhalb einer CALL-DML-Transaktion auch bestimmte SQL-Anweisungen angegeben werden. Diese Erweiterung ist nur beim independent DBH nutzbar. Anwendungen mit dem linked-in DBH dürfen auch weiterhin nur reine CALL-DML- oder SQL-Transaktionen enthalten. Weitere Informationen dazu finden Sie im Abschnitt „Transaktion im Anwenderprogramm“ auf Seite 200 und im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“.

4.6 Programmeinbettung von SQL

Um von einem Programm aus auf die Datenbank zugreifen zu können, gibt es programmiersprachenspezifische Schnittstellen, die es ermöglichen, SQL-Anweisungen in ein Programm einzubinden. SESAM/SQL bietet eine Schnittstelle für die Programmiersprache COBOL.

4.6.1 ESQL-Programm

Ein ESQL-Programm ist ein Programm in einer Programmiersprache, in das zusätzlich SQL-Anweisungen eingebettet sind. Diese Programmiersprache eines ESQL-Programms wird auch Wirtssprache (host language) genannt. SESAM/SQL kennt ESQL-Programme für die Wirtssprache COBOL (siehe Handbuch „ESQL-COBOL für SESAM/SQL-Server“).

Um die Anweisungen der Wirtssprache von den SQL-Anweisungen unterscheiden zu können, werden Anfang und Ende einer SQL-Anweisung im ESQL-Programm jeweils markiert. Ein ESQL-Precompiler trennt Anweisungen der Wirtssprache und SQL-Anweisungen und erzeugt ein SQL-Objektmodul, in dem die SQL-Anweisungen zusammengefasst sind. Das verbleibende COBOL-Programm, das keine SQL-Anweisungen mehr enthält, wird mit dem COBOL-Compiler übersetzt. Das entstehende Modul wird mit dem SQL-Modul und dem Laufzeitsystem des COBOL-Compilers zu einem Lademodul gebunden.

In ESQL-COBOL-Programmen werden eingebettete SQL-Anweisungen mit dem Schlüsselwort „EXEC SQL“ eingeleitet und mit „END-EXEC“ beendet. Ausführbare SQL-Anweisungen können überall da im Programm auftreten, wo eine ausführbare Anweisung der Wirtssprache auftreten kann. Ausführbar sind alle SQL-Anweisungen bis auf die Anweisungen DECLARE CURSOR und WHENEVER. National-Datentypen können nur mit entsprechender BS2000-Systemumgebung inklusive entsprechendem ESQL-Precompiler und COBOL-Compiler verwendet werden, siehe Seite 28 .

4.6.2 Benutzervariablen

SQL-Anweisungen können Variablen der Wirtssprache, sogenannte Benutzervariablen, enthalten. Mit Benutzervariablen ist es möglich, Daten aus der Datenbank in das Programm der Wirtssprache zu übertragen und dort weiterzuverarbeiten und umgekehrt Daten in die Datenbank zu übertragen. In SQL-Anweisungen müssen alle Benutzervariablen mit einem vorangestellten Doppelpunkt gekennzeichnet sein. Benutzervariablen und Spalten können denselben Namen haben.

Benutzervariablen müssen vor ihrer Verwendung innerhalb einer DECLARE SECTION definiert werden. Eine DECLARE SECTION beginnt in ESQL-COBOL mit „EXEC SQL BEGIN DECLARE SECTION END-EXEC“ und endet mit „EXEC SQL END DECLARE SECTION END-EXEC“. Ein ESQL-Programm kann beliebig viele DECLARE SECTIONs enthalten.

DECLARE SECTIONs dürfen nur an Stellen stehen, an denen Variablendeklarationen in der Programmiersprache erlaubt sind. Wird eine Benutzervariable in einer ausführbaren SQL-Anweisung verwendet, so muss nach den Regeln der Wirtssprache eine Referenz auf die Deklaration an dieser Stelle erlaubt sein.

Bei der Definition der Benutzervariablen muss auf die Verträglichkeit von Datentypen der Wirtssprache und SQL-Datentypen geachtet werden. Auf die Datentyp-Verträglichkeit ist nicht nur bei der Übertragung von Werten in die Datenbank oder von der Datenbank zu achten, sondern auch bei der Verwendung von Benutzervariablen in Ausdrücken.

Indikatorvariablen

Indikatorvariablen sind Benutzervariablen, mit denen geprüft werden kann, ob bei der Übertragung eines alphanumerischen Werts oder eines National-Werts in eine Benutzervariable Informationsverlust aufgetreten ist und ob eine Spalte den NULL-Wert enthält. Indikatorvariablen dienen auch dazu, NULL-Werte in die Datenbank zu übertragen.

Wenn ein alphanumerischer Wert oder ein National-Wert ungekürzt in eine Benutzervariable übertragen wurde, dann hat die Indikatorvariable den Inhalt „0“. War die Länge des zu übertragenden Werts größer als die Länge der Benutzervariable, dann enthält die Indikatorvariable die tatsächliche Länge des zu übertragenden Werts.

Soll der NULL-Wert aus der Datenbank in eine Benutzervariable übertragen werden, dann erhält die Indikatorvariable den Wert „-1“ und die Benutzervariable bleibt ungeändert. Soll der NULL-Wert aus einer Benutzervariablen in die Datenbank übertragen werden, dann muss die zugehörige Indikatorvariable mit einem negativen Wert belegt werden.

Eine Indikatorvariable wird einer Benutzervariablen in einer SQL-Anweisung zugeordnet, indem die Indikatorvariable direkt hinter der Benutzervariablen, wahlweise mit dem Schlüsselwort INDICATOR versehen, angegeben wird.

Indikatorvariablen können angegeben werden

- in der RETURN INTO- oder VALUES-Klausel der INSERT-Anweisung

- in der VALUES-Klausel der MERGE-Anweisung
- in der INTO-Klausel der SELECT- oder der FETCH-Anweisung
- in der SET-Klausel von UPDATE- oder MERGE-Anweisungen
- in der USING- und INTO-Klausel der EXECUTE-Anweisung
- in der USING-Klausel der OPEN-Anweisung
- in Ausdrücken, z.B. in DELETE- oder DECLARE CURSOR-Anweisungen.

4.6.3 Erfolgskontrolle und Fehlerbehandlung

Nach der Ausführung einer SQL-Anweisung sollte im ESQL-Programm abgefragt werden, ob die SQL-Anweisung erfolgreich ausgeführt wurde oder ob die Bearbeitung zu einem Fehler führte. Zu diesem Zweck stellt die ESQL-Schnittstelle die Benutzervariablen SQLSTATE und SQLCODE zur Verfügung. In diese werden die Informationen über die Ausführung einer SQL-Anweisung abgelegt. SQLSTATE enthält einen SQL-Statuscode, SQLCODE einen SQL-Returncode, der die gleiche Funktion wie der SQL-Statuscode hat, aber nur noch aus Gründen der Kompatibilität zu früheren SESAM/SQL-Versionen unterstützt wird.

Die fünfstelligen alphanumerischen SQL-Statuscodes bestehen aus einem zweistelligen Teil für die Klasse des Statuscodes und einem dreistelligen für eine Unterklasse. Eine fehlerfreie Ausführung einer SQL-Anweisung wird durch SQLSTATE-Werte mit den Klassen „erfolgreiche Ausführung“, „Warnung“ oder „keine Daten“ angezeigt. Nach einer fehlerhaften Bearbeitung enthält SQLSTATE Informationen über die Art des Fehlers.

Die numerischen Werte von SQLCODE sind: 0 für eine erfolgreiche Ausführung, 100, wenn eine Tabelle leer ist oder das Tabellenende erreicht wurde, und 10 oder 50, wenn eine Warnung aufgetreten ist. Im Fehlerfall liefert SQLCODE einen negativen, für die Fehlerbedingung spezifischen Wert.

Um zu prüfen, ob eine SQL-Anweisung korrekt ausgeführt wurde, und um in Abhängigkeit vom zurückgelieferten SQL-Status- oder SQL-Returncode Aktionen im Wirtsprogramm zu steuern, gibt es zwei Möglichkeiten. Einmal kann der ESQL-Programmierer den Programmablauf mit Anweisungen der Wirtssprache abhängig vom Wert von SQLSTATE (oder SQLCODE) steuern. Eine andere Möglichkeit besteht in der Verwendung der SQL-Anweisung WHENEVER.

4.6.4 SQL-Datenmanipulation mit Cursor

Um innerhalb eines ESQL-Programms Sätze einer Ergebnistabelle bearbeiten zu können, verwendet der Anwender in SQL einen Cursor. Ein Cursor ist ein Zeiger innerhalb einer besonderen Ergebnistabelle, der sogenannten Cursortabelle, mit dem der Anwender auf die Sätze der Tabelle zugreifen kann.

Die Cursortabelle wird durch einen Abfrage-Ausdruck definiert, der bei der Definition des Cursors mit DECLARE CURSOR angegeben werden kann. Dieser Abfrage-Ausdruck wird erst dann ausgewertet, wenn der Cursor mit einer OPEN-Anweisung geöffnet wird. Enthält der Abfrage-Ausdruck Benutzervariablen, dann werden die zu diesem Zeitpunkt aktuellen Werte dieser Benutzervariablen herangezogen. Die Sätze können nun gelesen, oder, bei einem änderbaren Cursor, geändert oder gelöscht werden, bis der Cursor mit einer CLOSE-Anweisung wieder geschlossen wird.

Die SQL-Anweisung FETCH positioniert einen geöffneten Cursor auf einen Satz der Ergebnistabelle und macht diesen Satz zum aktuellen Satz.

Die UPDATE... CURRENT-Anweisung ändert den aktuellen Satz, auf den der Cursor zeigt.

Die DELETE... CURRENT-Anweisung löscht den Satz, auf den der Cursor zeigt.

Mit der STORE- und RESTORE-Anweisung wird verhindert, dass am Ende einer Transaktion die Position des Cursors verlorengeht. Die aktuelle Cursorposition bleibt nur erhalten, wenn sie vor dem Schließen mit STORE gesichert wird. RESTORE stellt einen mit STORE gesicherten Cursor wieder her.

Cursor mit Schubmodus (siehe auch Seite 153)

Durch Angabe des Pragmas PREFETCH in der DECLARE CURSOR-Anweisung kann ein Cursor mit sogenanntem Schubmodus definiert werden. Im Schubmodus veranlasst die erste Ausführung der FETCH NEXT-Anweisung, dass mehrere Sätze in einen Zwischenpuffer gelesen werden. Der Anwender erhält mit der ersten Ausführung der FETCH-Anweisung nur den ersten übertragenen Satz. Die nachfolgenden Ausführungen der FETCH-Anweisung übertragen jeweils einen Satz aus dem Zwischenpuffer (ohne erneute Task-Task-Kommunikation), bis der Zwischenpuffer leer ist. Eine weitere Ausführung der FETCH-Anweisung führt dann wieder zur Übertragung mehrerer Sätze. Durch diesen Schubmodus kann die Bearbeitung eines Cursors wesentlich beschleunigt werden. Der Cursor mit Schubmodus darf stets nur mit ein und derselben FETCH-Anweisung angesprochen werden, d.h. derselben FETCH-Anweisung in einer Schleife oder einem Unterprogramm.

Die Größe des Zwischenpuffers kann mit der Anweisung PREFETCH-BUFFER (siehe Seite 276) festgelegt werden, die in der Konfigurationsdatei des Anwenderprogramms angegeben wird. Informationen über die Speicheranforderungen für den Zwischenpuffer und über die tatsächliche Nutzung liefert die Maske PREFETCH-BUFFERS des SESAM/SQL-Performance-Monitors SESMON (siehe Handbuch „Datenbankbetrieb“).

4.6.5 Dynamische SQL

Mit den bisher erwähnten SQL-Anweisungen kann ein ESQL-Programm erstellt werden, das einmal übersetzt wird und dann beliebig oft ablaufen kann. Die Form jeder SQL-Anweisung ist zum Kompilierungszeitpunkt bekannt.

Um beispielsweise beliebige Abfragen einer Datenbank durchzuführen, könnte man versuchen, alle möglichen Abfragen in einem ESQL-Programm vorzusehen. Eine bestimmte Abfrage könnte dann vom ESQL-Programm analysiert und die zugehörige SQL-Anweisung im ESQL-Programm angesprungen werden. Ein solcher Versuch würde offensichtlich sehr bald an Grenzen stoßen. Um derartige Anwendungen durchführen zu können, sind Sprachmittel notwendig, mit denen man SQL-Anweisungen erst zur Laufzeit eines ESQL-Programms angeben kann.

Solche Anwendungen können mit Hilfe von dynamisch übersetzten, d.h. erst zur Laufzeit übersetzten SQL-Anweisungen realisiert werden. Zum Übersetzen und Ausführen von dynamisch übersetzbaren SQL-Anweisungen gibt es spezielle Anweisungen. Sie werden „Anweisungen der dynamischen SQL“ oder kurz „dynamische SQL“ genannt. Sie umfassen neben den SQL-Anweisungen der dynamischen SQL auf Seite 109 noch SQL-Anweisungen für einen dynamischen, d.h. erst zur Laufzeit definierten Cursor. Im Unterschied zur dynamischen SQL werden SQL-Anweisungen, die zum Kompilierungszeitpunkt festliegen müssen, als statische SQL bezeichnet.

Die folgende Übersicht zeigt die zunehmenden Freiheitsgrade, die durch die verschiedenen Sprachmittel der dynamischen SQL erreicht werden können.

SQL-Sprachmittel		**Anzahl der Parameter zum Übersetzungszeitpunkt bekannt**	**Datentyp der Parameter zum Übersetzungszeitpunkt bekannt**	**Wert des Parameters**
Statische SQL		ja	ja	durch Benutzervariablen übergebbar
Dynamische SQL	EXECUTE IMMEDIATE	keine Parameter		
	PREPARE EXECUTE mit Benutzervariablen	ja	ja	Platzhalter als Ersatz für Eingabeparameter
	PREPARE EXECUTE mit Deskriptorbereichen	nein	nein	

Tabelle 36: Zunehmende Flexibilität beim Einsatz von dynamischer SQL

Diesem Gewinn an Flexibilität steht ein erhöhter Aufwand für die Anwendungsprogrammierung bei Anwendungen mit dynamischer SQL gegenüber.

Der am weitesten reichende Mechanismus, mit dem Ein- und Ausgabeparameter von dynamisch übersetzten SQL-Anweisungen gesetzt und gelesen werden können, ist der SQL-Deskriptorbereich. Ein SQL-Deskriptorbereich dient als Schnittstelle zwischen dem Anwenderprogramm und der Datenbank für Eingabeparameter oder Ausgabeparameter. SQL-Anweisungen, die erst zur Laufzeit eines ESQL-Programms eingegeben werden, können variable Eingabeparameter enthalten, die im SQL-Deskriptorbereich abgelegt und dort abgefragt werden können. Es können ebenfalls Ergebnisse von Abfragen der Datenbank im SQL-Deskriptorbereich abgelegt und Informationen über Anzahl, Name, Datentyp, Werte usw. der Ergebnisspalten von dort gelesen und im ESQL-Programm berücksichtigt werden. Beispielsweise können mit Hilfe von Deskriptorbereichen Ergebnisse von beliebigen Abfragen der Datenbank im ESQL-Programm aufbereitet werden.

4.6.5.1 Dynamisches Vorbereiten und Ausführen von SQL-Anweisungen

Die dynamische SQL soll zunächst mit einem einfachen Beispiel eingeführt werden. In diesem Beispiel wird noch kein SQL-Deskriptorbereich verwendet. Im Beispiel werden die SQL-Anweisungen wie in ESQL-COBOL mit EXEC SQL und END-EXEC gekennzeichnet. Die Aktionen in der Wirtssprache des ESQL-Programms sind in Kursiv dargestellt.

Beispiel

In der Tabelle AUFTRAG soll für den Satz mit der Auftragsnummer 251 und der Kundennummer 105 der Auftragsstatus auf 3 geändert werden. In statischer SQL kann dies durch folgende Anweisung erreicht werden:

```
UPDATE auftrag SET astnr=3 WHERE anr=251 AND knr=105
```

Dieser Anweisung entspricht in dynamischer SQL folgende Anweisungsfolge:

1. *Deklaration der Benutzervariablen Sourcestmt und Bedingung*
   Sourcestmt='UPDATE auftrag SET astnr=3 WHERE anr=251'
2. *Einlesen des Texts 'AND knr=105' in Bedingung*
   Sourcestmt=Sourcestmt||Bedingung
3. `EXEC SQL EXECUTE IMMEDIATE :Sourcestmt END_EXEC`

Erklärung

1. *Sourcestmt* ist eine Benutzervariable, die im ESQL-Programm deklariert ist und der eine alphanumerische Zeichenkette mit der UPDATE-Anweisung zugewiesen wird.

2. Die UPDATE-Anweisung wird durch eine Bedingung erweitert, indem der Text in *Sourcestmt* mit dem zur Laufzeit eingelesenen Text in *Bedingung* verkettet wird. Während in statischer SQL der Anweisungstext der UPDATE-Anweisung zum Übersetzungszeitpunkt des Programms festliegt, braucht hier die Anweisung erst zur Laufzeit des Programms festgelegt zu werden.
3. Die EXECUTE IMMEDIATE-Anweisung prüft und übersetzt die Anweisung und führt sie aus.

Beispiel

Eine allgemeinere Form der obigen UPDATE-Anweisung kann mit statischer SQL durch die folgende Anweisung erreicht werden:

```
UPDATE auftrag SET astnr=:ASTNR WHERE anr=:ANR AND knr=105
```

Hier werden die Werte für ASTNR und ANR durch zwei Benutzervariablen ersetzt, deren Werte erst zur Laufzeit des Programms festgelegt werden. Dieser Anweisung entspricht in dynamischer SQL folgende Anweisungsfolge:

1. *Deklaration der Benutzervariablen SOURCESTMT, BEDINGUNG, HOSTVAR1 und HOSTVAR2*
   sourcestmt='UPDATE auftrag SET astnr= ? WHERE anr= ?'
2. *Einlesen des Texts 'AND knr=105' in BEDINGUNG*
   sourcestmt=sourcestmt||bedingung
3. `EXEC SQL PREPARE dynstmt FROM :SOURCESTMT END-EXEC`
4. *Einlesen der Werte 3 für HOSTVAR1 und 251 für HOSTVAR2*

   ```
   EXEC SQL EXECUTE dynstmt USING :HOSTVAR1,:HOSTVAR2 END-EXEC
   ```

Erklärung

1. Die UPDATE-Anweisung enthält in der WHERE-Bedingung statt der Werte für ASTNR und ANR zwei Platzhalter, die durch Fragezeichen symbolisiert werden.
2. Die UPDATE-Anweisung wird durch eine Bedingung erweitert, indem der Text in SOURCESTMT mit dem zur Laufzeit eingelesenen Text in BEDINGUNG verkettet wird.
3. Diese dynamisch übersetzbare UPDATE-Anweisung wird mit der folgenden PREPARE-Anweisung übersetzt. DYNSTMT bezeichnet die dynamisch übersetzte SQL-Anweisung. Unter diesem Namen kann die Anweisung in einer EXECUTE-Anweisung angesprochen werden.

4. Vor der Ausführung der EXECUTE-Anweisung werden Werte für die im ESQL-Programm deklarierten Benutzervariablen HOSTVAR1 und HOSTVAR2 eingelesen. Diese Werte sollen die Platzhalter „?“ in der UPDATE-Anweisung ersetzen. Die EXECUTE-Anweisung führt nun die mit PREPARE übersetzte Anweisung DYNSTMT aus.

Beispiel

Es wird der Cursor CUR_LEISTUNG für die Tabelle LEISTUNG definiert. Die Cursorbeschreibung soll zur Laufzeit des Programms erzeugt werden. Der Anwender kann die Auftragsnummer und die Bedingung eingeben, die für die auszugebenden Leistungen gelten sollen.

1. *Deklaration der Benutzervariablen ANREINGABE, BEDINGUNG und BESCHREIBUNG. Außerdem wird die SQL-Variable AUFTRAGSNR definiert, die den Wert der Auftragsnummer aus ANREINGABE aufnimmt.*
2. Cursor CUR_LEISTUNG definieren:

```
EXEC SQL
     DECLARE cur_leistung CURSOR FOR CURBESCHREIBUNG
END-EXEC
```

3. *Eingabe der Werte für ANREINGABE und BEDINGUNG durch den Anwender zur Laufzeit des Programms.*
4. *Erzeugen der Beschreibung BESCHREIBUNG:*

```
STRING "SELECT * FROM leistung WHERE anr = ? AND " bedingung
```

5. Cursorbeschreibung erzeugen:

```
EXEC SQL
     PREPARE CURBESCHREIBUNG FROM :BESCHREIBUNG
END-EXEC
```

6. Cursor CUR_LEISTUNG öffnen. Der Platzhalter „?“ in der Cursorbeschreibung wird dabei mit der vom Anwender gewählten Auftragsnummer besetzt:

```
EXEC SQL
     OPEN cur_leistung USING :AUFTRAGSNR
END-EXEC
```

Die in den Beispielen verwendeten Anweisungen EXECUTE IMMEDIATE, PREPARE und EXECUTE werden im Folgenden näher dargestellt. Zunächst wird beschrieben, welche Anweisungen dynamisch übersetzbar sind.

Dynamisch übersetzbare Anweisungen

Dynamisch übersetzbar sind folgende Anweisungen (zur Klassifikation der Anweisungen siehe Seite 107):

- SQL-Anweisungen zur Schemadefinition und -verwaltung
- SQL-Anweisungen zur Gestaltung und Verwaltung von Routinen
- SQL-Anweisungen zum Abfragen und Ändern von Daten:
  - SELECT (ohne INTO-Klausel)
  - UPDATE
  - DELETE
  - INSERT (ohne RETURN INTO-Klausel)
  - MERGE
  - CALL
- SQL-Anweisungen zur Transaktionsverwaltung
- SQL-Anweisungen zur Sessionsteuerung
- SQL-Anweisungen zur Verwaltung der Speicherstruktur
- SQL-Anweisungen zur Verwaltung von Benutzereinträgen
- Utility-Anweisungen

Außerdem sind Cursorbeschreibungen dynamisch übersetzbar.

Nicht dynamisch übersetzbar sind

- die SQL-Anweisungen der dynamischen SQL
- die SQL-Anweisungen für das Arbeiten mit einem Cursor:
  - DECLARE CURSOR
  - OPEN
  - FETCH
  - CLOSE
  - STORE
  - RESTORE
- die SQL-Anweisungen INCLUDE und WHENEVER.

EXECUTE IMMEDIATE-Anweisung

Mit der Anweisung EXECUTE IMMEDIATE wird eine dynamisch formulierte Anweisung in einem Schritt vorbereitet und ausgeführt. EXECUTE IMMEDIATE entspricht also einer PREPARE-Anweisung mit unmittelbar darauffolgender EXECUTE-Anweisung. Allerdings bleibt die Anweisung nicht vorbereitet und kann nicht mit EXECUTE nochmals ausgeführt werden.

Innerhalb des Anweisungstexts dürfen keine Benutzervariablen und keine Fragezeichen als Platzhalter für unbekannte Werte verwendet werden.

PREPARE-Anweisung

PREPARE bereitet eine dynamisch formulierte Anweisung oder die Cursorbeschreibung eines dynamischen Cursors für die spätere Ausführung vor.

Eine mit PREPARE vorbereitete Anweisung wird mit EXECUTE ausgeführt.

Innerhalb des Anweisungstexts dürfen keine Benutzervariablen verwendet werden. Platzhalter für noch unbekannte Eingabewerte werden durch ein Fragezeichen angegeben.

Platzhalter

Die in der Anweisungsvariablen enthaltene dynamisch übersetzbare Anweisung bzw. dynamisch übersetzbare Cursorbeschreibung darf keine Benutzervariablen enthalten. Stattdessen dürfen darin Platzhalter für Eingabewerte stehen, die durch Fragezeichen dargestellt werden. Diese Platzhalter werden in der USING-Klausel einer späteren EXECUTE- oder OPEN-Anweisung mit Werten versorgt.

Für jeden Platzhalter wird ein SQL-Datentyp bestimmt. Es gibt einige Regeln, die die Verwendung von Platzhaltern in *anweisungsvariable* einschränken. Ein Platzhalter ist nur in einem Kontext erlaubt, in dem der Datentyp des Platzhalters eindeutig bestimmbar ist.

Platzhalter sind deshalb nicht erlaubt

- als Element einer SELECT-Liste in der Form „?“ (dagegen ist z.B. „SELECT ?+1“ erlaubt)
- als Operand eines einstelligen Operators
- als Argument einer Mengenfunktion
- in einer Abfrage auf den NULL-Wert
- wenn beide Operanden eines zweistelligen Operators oder eines Vergleichsoperators Platzhalter sind
- wenn bei einer Bereichsabfrage mit BETWEEN, in der der erste Operand ein Platzhalter ist, einer der beiden anderen Operanden ein Platzhalter ist

- als erster Operand bei einem Mustervergleich mit LIKE
- wenn bei einem Vergleich eines Ausdrucks mit einer Menge von Werten sowohl der zu vergleichende Ausdruck als auch alle Werte in der Liste nach IN Platzhalter sind
- wenn in einem CASE-Ausdruck alle Operanden Platzhalter sind. Enthält der CASE-Ausdruck eine oder mehrere THEN- bzw. ELSE-Klauseln, dürfen zusätzlich nicht alle Operanden dieser Klauseln Platzhalter sein.
- als Operanden der Zeichenkettenfunktionen LOWER und UPPER
- als erster Operand (*zeichen*) und/oder als zweiter Operand (*ausdruck*) der Zeichenkettenfunktion TRIM (z.B. TRIM (TRAILING FROM ?))
- als erster Operand der Zeichenkettenfunktion SUBSTRING (z.B. SUBSTRING ? FROM 1 FOR 5))
- wenn beide Operanden der numerischen Funktion POSITION Platzhalter sind.

EXECUTE-Anweisung

EXECUTE führt eine mit PREPARE vorbereitete Anweisung aus. Platzhalter für Eingabewerte in der dynamisch formulierten Anweisung werden durch aktuelle Werte ersetzt.

Ist die Anweisung eine SELECT-Anweisung, werden die Spaltenwerte des Ergebnissatzes in Benutzervariablen oder einem SQL-Deskriptorbereich abgelegt.

Eine EXECUTE-Anweisung kann beliebig oft für eine mit PREPARE vorbereitete Anweisung ausgeführt werden.

Eine Anweisung kann mit EXECUTE nur in der Übersetzungseinheit ausgeführt werden, in der sie vorher mit PREPARE vorbereitet wurde.

In der EXECUTE-Anweisung können Benutzervariablen oder SQL-Deskriptorbereiche für die Ein- und Ausgabewerte verwendet werden. Die Verwendung von SQL-Deskriptorbereichen wird im nächsten Abschnitt beschrieben.

4.6.5.2 SQL-Deskriptorbereich

Bei den bisher beschriebenen Anweisungen der dynamischen SQL liegen Anzahl und Datentyp der Platzhalter und Ergebnisspalten einer dynamisch übersetzbaren SQL-Anweisung oder dynamischen Cursorbeschreibung zum Übersetzungszeitpunkt des ESQL-Programms fest. Um beliebige Eingabeparameter verwenden zu können und um beliebige Ergebnisse von Abfragen im ESQL-Programm verarbeiten zu können, stehen in SQL sogenannte Deskriptorbereiche zur Verfügung.

Ein Deskriptorbereich ist ein Speicherbereich, in den Werte und Informationen über Anzahl und Datentyp von Eingaben (Platzhaltern) und Ausgaben (Ergebnisspalten) von dynamisch übersetzbaren SQL-Anweisungen oder Cursorbeschreibungen eingetragen werden.

Jeder Deskriptorbereich besteht aus einer Anzahl von Einträgen. Ein Eintrag wird für jede einfache Spalte oder für jeden einfachen Wert bzw. im Fall von multiplen Spalten oder Aggregaten für jedes Spaltenelement oder jede Ausprägung angelegt. Jeder Eintrag ist in Felder unterteilt. Die Werte in diesen Feldern können mit DESCRIBE oder SET DESCRIPTOR gesetzt und mit GET DESCRIPTOR gelesen werden. Je nach Art des Eintrags sind nur bestimmte Felder mit Werten belegt. Für jeden Deskriptorbereich gibt es ein Feld COUNT, das die Anzahl der Einträge eines Deskriptorbereichs enthält.

Der Deskriptorbereich enthält folgende Felder für einen Eintrag:

Schlüsselwort	**Bedeutung**	**kann im SQL-Programm gesetzt werden**
COUNT	Anzahl der Einträge eines Deskriptorbereichs	ja
TYPE	Datentyp des Eintrags: -42 NVARCHAR -31 NCHAR 1 CHAR 2 NUMERIC 3 DECIMAL 4 INTEGER 5 SMALLINT 6 FLOAT 7 REAL 8 DOUBLE PRECISION 9 Zeitdatentyp 12 VARCHAR	ja
LENGTH	Länge bzw. maximale Länge in Zeichen bei alphanumerischen Datentypen und Zeit-Datentypen oder Länge bzw. maximale Länge in Code Units bei National-Datentypen	ja
OCTET_LENGTH	Maximaler Speicherplatzbedarf in Bytes bei alphanumerischen Datentypen, National-Datentypen, numerischen Datentypen und Zeit-Datentypen	nein, nur gelesen
PRECISION	Nur für numerischen Datentypen sowie TIME und TIMESTAMP: Gesamtstellenzahl in Dezimalstellen (bei NUMERIC, DECIMAL, TIME, TIMESTAMP) bzw. Binärstellen (sonst)	ja

Tabelle 37: Felder eines Deskriptorbereichs (Teil 1 von 2)

Schlüsselwort	Bedeutung	kann im SQL-Programm gesetzt werden
SCALE	Nur für Ganz- oder Festpunktzahlen: Anzahl der Nachkommastellen	ja
DATETIME_ INTERVAL_CODE	Nur bei Zeitdatentypen (TYPE =9): 1 DATE 2 TIME 3 TIMESTAMP	ja
REPETITIONS	Bei multiplen Spalten bzw. Aggregaten: enthält beim ersten Eintrag die Anzahl der Komponenten, bei den folgenden Einträgen jeweils 1. Bei einfachen Spalten: nur den Wert 1	ja
NULLABLE	1 Ausgabewert kann der NULL-Wert sein 0 sonst	nein
INDICATOR	Bei Deskriptorbereich für Ergebnisspalten: 0 DATA enthält den gelesenen Wert -1 der NULL-Wert wurde gelesen >0 Originallänge eines alphanumerischen Werts bzw. eines National-Werts bei Informationsverlust nach der Übertragung Bei Deskriptorbereich für Platzhalter: <0 Wert ist der NULL-Wert 0 sonst	ja
DATA	Wert eines Eintrags. Der Datentyp wird durch die Einträge in TYPE, LENGTH, PRECISION, SCALE und DATETIME_INTERVAL_CODE bestimmt	ja
NAME	Spaltenname	nein
UNNAMED	1 NAME enthält Spaltenname 0 sonst	nein

Tabelle 37: Felder eines Deskriptorbereichs (Teil 2 von 2)

Das Feld COUNT ist pro Deskriptorbereich einmal vorhanden. Der Datentyp von NAME ist CHAR(n) oder VARCHAR(n) mit n ≥ 128, der Datentyp von DATA wird durch die Einträge in TYPE, LENGTH, PRECISION und SCALE bestimmt. Die restlichen Felder haben den Datentyp SMALLINT.

Der Anwender muss sich um die interne Struktur des SQL-Deskriptorbereichs nicht kümmern, sondern kann die Felder des Deskriptorbereichs mit SQL-Anweisungen ansprechen. Je nach Anweisung werden nur bestimmte Felder des Deskriptorbereichs angesprochen.

Für das Arbeiten mit einem SQL-Deskriptorbereich gibt es eine Reihe von SQL-Anweisungen und Varianten der EXECUTE-, OPEN- und FETCH-Anweisung:

- Deskriptorbereich anlegen
  Ein Deskriptorbereich muss zunächst mit ALLOCATE DESCRIPTOR angelegt werden. Mit ALLOCATE DESCRIPTOR wird die maximale Anzahl der Einträge im Deskriptorbereich angegeben. Die Einträge des Deskriptorbereichs sind nach ALLOCATE DESCRIPTOR noch undefiniert.
- Anzahl und Datentypen von Platzhaltern beschreiben
  Anzahl und Datentypen der Platzhalter einer dynamisch übersetzten SQL-Anweisung oder Cursorbeschreibung können mit DESCRIBE INPUT bestimmt und in einem Deskriptorbereich abgelegt werden.
- Werte für Platzhalter übergeben
  Die Werte für die Platzhalter einer dynamisch übersetzten Anweisung werden bei EXECUTE... USING SQL DESCRIPTOR aus einem Deskriptorbereich entnommen. Die Werte für die Platzhalter einer dynamisch übersetzten Cursorbeschreibung werden bei OPEN... USING SQL DESCRIPTOR aus einem Deskriptorbereich entnommen. In beiden Fällen müssen die gewünschten Werte vorher mit SET DESCRIPTOR gesetzt werden.
- Anzahl und Datentypen von Ergebnisspalten beschreiben
  Anzahl und Datentypen der Ergebnisspalten einer dynamisch übersetzten SELECT-Anweisung oder Cursorbeschreibung können mit DESCRIBE OUTPUT bestimmt und in einem Deskriptorbereich abgelegt werden.
- Spaltenwerte eines Ergebnissatzes ablegen
  Die Spaltenwerte des Ergebnissatzes einer dynamisch übersetzten SELECT-Anweisung werden bei EXECUTE... INTO SQL DESCRIPTOR in einen Deskriptorbereich abgelegt.
  Die Spaltenwerte eines Satzes der Ergebnistabelle eines dynamischen Cursors werden bei FETCH... INTO SQL DESCRIPTOR in einen Deskriptorbereich abgelegt.
- Deskriptorbereich ändern
  Die Anzahl der Einträge oder der Inhalt eines Eintrags des Deskriptorbereichs können mit SET DESCRIPTOR geändert werden.
- Deskriptorbereich abfragen
  Die Anzahl der Einträge oder der Inhalt von Feldern eines Eintrags im Deskriptorbereich können mit GET DESCRIPTOR gelesen werden.
- Deskriptorbereich freigeben
  Der Speicherplatz, den ein Deskriptorbereich belegt, kann mit DEALLOCATE DESCRIPTOR wieder freigegeben werden.

4.6.5.3 Dynamische Cursor

Cursor, die ohne Cursorbeschreibung vereinbart werden, nennt man dynamische Cursor.

Ein dynamischer Cursor wird mit DECLARE CURSOR vereinbart.
Mit PREPARE wird ein dynamischer Cursor mit einer dynamisch übersetzbaren Cursorbeschreibung verbunden. Wie bei einem statischen Cursor kann der Anwender mit OPEN-, FETCH- und CLOSE-Anweisungen auf Sätze einer Tabelle zugreifen und mit UPDATE CURRENT- und DELETE CURRENT-Anweisungen Sätze einer Tabelle ändern oder löschen.

Wie bei Anweisungen der dynamischen SQL kann durch dynamische Cursor ein zunehmender Grad an Flexibilität erreicht werden, der auf der anderen Seite einen erhöhten Programmieraufwand erfordert:

SQL-Sprachmittel		**Anzahl der Parameter zum Übersetzungszeitpunkt bekannt**	**Datentyp der Parameter zum Übersetzungszeitpunkt bekannt**	**Wert der Parameter**
Statische Cursor		ja	ja	durch Benutzervariablen übergebbar
Dynamische Cursor DECLARE CURSOR PREPARE OPEN... USING FETCH ... INTO CLOSE	mit Benutzervariablen	ja	ja	Platzhalter als Ersatz für Eingabeparameter
	mit Deskriptorbereichen	nein	nein	

Tabelle 38: Zunehmende Flexibilität beim Einsatz dynamischer Cursor

Enthält eine dynamisch übersetzbare Cursorbeschreibung Platzhalter, müssen die zugehörigen Werte über Benutzervariablen oder einen zuvor belegten Deskriptorbereich in der USING-Klausel der OPEN-Anweisung zur Verfügung gestellt werden.

Mit Hilfe eines SQL-Deskriptorbereichs können für eine dynamisch übersetzte Cursorbeschreibung

- mit DESCRIBE INPUT die Datentypen der Platzhalter bestimmt werden
- mit DESCRIBE OUTPUT Anzahl und Datentypen der Ergebnisspalten bestimmt und in einem Deskriptorbereich abgelegt werden
- mit OPEN USING die Werte für die Platzhalter aus einer Benutzervariablen oder einem Deskriptorbereich entnommen werden
- mit FETCH INTO die Werte eines Satzes der Ergebnistabelle in einen Deskriptorbereich abgelegt werden.

Dynamischer Cursor mit Schubmodus

Wie für einen statischen Cursor (siehe Seite 141) lässt sich auch für einen dynamischen Cursor ein Schubmodus definieren. Dazu wird das Pragma PREFETCH in der Cursorbeschreibung des dynamischen Cursors angegeben.

Wird eine FETCH-Anweisung mit einem Deskriptorbereich verwendet, dann darf dieser Deskriptorbereich nicht in SET DESCRIPTOR-, DESCRIBE- oder DEALLOCATE DESCRIPTOR-Anweisungen angesprochen werden, solange der Schubmodus aktiv ist. Außerdem gelten für einen dynamischen Cursor dieselben Einschränkungen wie für einen statischen Cursor mit Schubmodus.

Beispiel

```
DECLARE dyn_cursor CURSOR FOR dynstmt
```

Einlesen von SOURCESTMT:

```
'--%PRAGMA PREFETCH 10
SELECT knr, firma FROM kunde WHERE land = 'D  ''

PREPARE dynstmt FROM :SOURCESTMT

OPEN dyn_cursor
```

Schleife bis SQLSTATE = 00200:

```
FETCH NEXT FROM dyn_cursor INTO :KNR, :FIRMA
```

Bei der Cursorbeschreibung des dynamischen Cursors DYN_CURSOR wird das Pragma PREFETCH mit dem Blockungsfaktor 10 angegeben. Der Anwender erhält mit dem ersten FETCH NEXT den ersten Satz der Cursortabelle; gleichzeitig werden die 9 nächsten Sätze in den Zwischenpuffer gelesen. Der Cursor mit Schubmodus darf stets nur mit ein und derselben FETCH-Anweisung angesprochen werden, d.h. derselben FETCH-Anweisung in einer Schleife oder einem Unterprogramm. Mit der erneuten Ausführung dieser FETCH-Anweisung wird der zweite Satz des Cursors beschleunigt aus dem Zwischenpuffer gelesen.

4.6.5.4 Voreinstellungen in dynamischen Anweisungen festlegen

Mit der Anweisung SET CATALOG kann der Anwender einen voreingestellten Datenbanknamen festlegen, mit dem einfache Schemanamen in dynamisch übersetzbaren SQL-Anweisungen qualifiziert werden.

SET SCHEMA legt einen voreingestellten Schemanamen fest, mit dem einfache Namen von Tabellen, Integritätsbedingungen und Indizes in dynamisch übersetzbaren SQL-Anweisungen qualifiziert werden.

Der voreingestellte Datenbank- bzw. Schemaname wird aus dem Wert eines alphanumerischen Literals oder dem Wert einer Benutzervariablen von alphanumerischem Datentyp abgeleitet. Der Datenbank- bzw. Schemaname gilt bis zur nächsten SET CATALOG- bzw. SET SCHEMA-Anweisung. Ein voreingestellter Datenbank- und ein voreingestellter Schemaname können auch mit einer SET SCHEMA-Anweisung durch Angabe eines qualifizierten Schemanamens festgelegt werden.

4.6.5.5 Dynamische SQL mit Deskriptorbereichen

Beispiel

Eine beliebige SQL-Anweisung wird zur Laufzeit des Programms eingelesen. Es wird angenommen, dass Anzahl und Datentyp der Eingabewerte beim Schreiben des Programms noch nicht bekannt sind. Sowohl Anzahl und Datentypen der Eingabewerte als auch die Spaltenwerte der Ergebnissätze werden mit Hilfe von DESCRIBE ermittelt.

Im Beispiel werden die SQL-Anweisungen wie in ESQL-COBOL mit EXEC SQL und END-EXEC gekennzeichnet. Die Aktionen in der Wirtssprache des ESQL-Programms sind in Kursiv dargestellt.

Die Fehlerbehandlung wird im vorliegenden Beispiel nur an einigen Stellen dargestellt. Ein vollständiges Beispiel erfordert natürlich nach jeder ausführbaren SQL-Anweisung eine Abfrage des SQLSTATE und eine entsprechende Fehlerbehandlung.
Es wird vorausgesetzt, dass keine multiplen Spalten vorliegen.

Deklaration von Benutzervariablen H_COUNT, H_TYPE, H_LENGTH, H_NAME, usw., die die Werte für COUNT, TYPE, LENGTH, NAME, usw. in den Feldern des Deskriptorbereichs enthalten bzw. aufnehmen sollen.

Deklaration von Benutzervariablen, die jeweils einem SQL-Datentyp entsprechen: H_CHARACTER, H_NUMERIC1, H_NUMERIC2, usw. für verschiedene Wertebereiche, usw.

Deklaration der Benutzervariablen SOURCESTMT zur Aufnahme der dynamisch übersetzbaren Anweisung

1. `EXEC SQL DECLARE dyn_cursor SCROLL CURSOR FOR dynstmt END-EXEC`
2. `EXEC SQL ALLOCATE DESCRIPTOR GLOBAL 'Placehold' END-EXEC`
   `EXEC SQL ALLOCATE DESCRIPTOR GLOBAL 'Results' END-EXEC`
3. *Einlesen von SOURCESTMT*

   `EXEC SQL PREPARE dynstmt FROM :SOURCESTMT END-EXEC`
4. a) `EXEC SQL DESCRIBE INPUT dynstmt`
      `USING SQL DESCRIPTOR GLOBAL 'Placehold' END-EXEC`

      Fehlerbehandlung, falls nicht SQLSTATE(1:2) = "00"

   b) `EXEC SQL DESCRIBE OUTPUT dynstmt USING SQL DESCRIPTOR GLOBAL 'Results'`
      `END-EXEC`

      Fehlerbehandlung, falls nicht SQLSTATE(1:2) = "00"
5. `EXEC SQL GET DESCRIPTOR GLOBAL 'Placehold' :H_COUNT = COUNT END-EXEC`
6. *Schleife von I=1 bis I=H_COUNT*

   ```
   EXEC SQL GET DESCRIPTOR GLOBAL 'Placehold' VALUE :I
   :H_TYPE =TYPE,
   :H_LENGTH = LENGTH, ..., END-EXEC
   ```
7. *Auswahl einer geeigneten Benutzervariablen entsprechend dem Datentyp des Platzhalters. Anpassen des Datentyps und Einlesen des Werts für den Platzhalter in den Deskriptorbereich, z.B.:*

   wenn :H_TYPE = 1 oder 12
   Einlesen des Werts für den Platzhalter in H_CHARACTER

   ```
   EXEC SQL SET DESCRIPTOR GLOBAL 'Placehold' VALUE :I TYPE = 1,
   LENGTH=256, INDICATOR=0, DATA=:H_CHARACTER END-EXEC
   ```

   wenn :H_TYPE =2 oder 3 und 10 bis 18 Vorkommastellen
   Einlesen des Werts für den Platzhalter in H_NUMERIC1

   ```
   EXEC SQL SET DESCRIPTOR GLOBAL 'Placehold' VALUE :I TYPE=2,
   PRECISION=18, SCALE=0, INDICATOR=0, DATA=:H_NUMERIC1 END-EXEC
   ```

wenn :H_TYPE =2 oder 3 und 1 bis 9 Vorkommastellen
Einlesen des Werts für den Platzhalter in H_NUMERIC2

```
EXEC SQL SET DESCRIPTOR GLOBAL 'Placehold' VALUE :I TYPE=2,
PRECISION=18, SCALE=9, INDICATOR=0, DATA=:H_NUMERIC2 END-EXEC
```

wenn :H_TYPE =2 oder 3 und 0 Vorkommastellen
Einlesen des Werts für den Platzhalter in H_NUMERIC3

```
EXEC SQL SET DESCRIPTOR GLOBAL 'Placehold' VALUE :I TYPE=2,
PRECISION=18, SCALE=18, INDICATOR=0, DATA=:H_NUMERIC3 END-EXEC
```

usw.

Schleifenende

8. `EXEC SQL GET DESCRIPTOR GLOBAL 'Results' :H_COUNT=COUNT END-EXEC`

   Wenn :H_COUNT=0

   ```
   EXEC SQL EXECUTE dynstmt USING SQL DESCRIPTOR GLOBAL 'Placehold'
   END-EXEC
   ```

   Sprung nach (12)

9. *sonst*

   Schleife von I=1 bis I=H_COUNT

   ```
   EXEC SQL GET DESCRIPTOR GLOBAL 'Results' VALUE :I
   :H_TYPE=TYPE
   :H_LENGTH=LENGTH ... END-EXEC
   ```

   Auswahl einer geeigneten Benutzervariablen entsprechend dem Datentyp der Spaltenwerte der Ergebnistabelle. Anpassen des Datentyps im Deskriptorbereich, analog dem Vorgehen (6) bei den Platzhaltern, z.B.:

   wenn :H_TYPE = 1 oder 12

   ```
   EXEC SQL SET DESCRIPTOR GLOBAL 'Results' VALUE :I TYPE = 1, LENGTH=256
   END-EXEC
   ```

   usw.

   Schleifenende

10. `EXEC SQL OPEN dyn_Cursor USING SQL DESCRIPTOR GLOBAL 'Placehold' END-EXEC`

11. *Ausführen der folgenden FETCH-Anweisung in einer Schleife, bis die Bedingung „Tabellenende“ auftritt*

```
EXEC SQL FETCH NEXT FROM dyn_cursor INTO SQL DESCRIPTOR GLOBAL
'Results' END-EXEC
```

Fehlerbehandlung, falls nicht SQLSTATE(1:2) = "00"

Schleife über alle Spalten des Ergebnissatzes:
Wenn :H_TYPE =1 oder 12

```
EXEC SQL GET DESCRIPTOR GLOBAL 'Results' VALUE :I
:H_NAME=NAME, :H_CHARACTER=DATA END-EXEC
```

wenn :H_TYPE =2 oder 3 und 10 bis 18 Vorkommastellen

```
EXEC SQL GET DESCRIPTOR GLOBAL 'Results' VALUE :I
:H_NAME=NAME, :H_NUMERIC1=DATA END-EXEC
```

usw.

Ausgabe von H_NAME und der entsprechenden Benutzervariable

Schleifenende

Schleifenende

12. `EXEC SQL CLOSE dyn_cursor END-EXEC`

13. `EXEC SQL DEALLOCATE DESCRIPTOR GLOBAL 'Placehold' END-EXEC`

```
EXEC SQL DEALLOCATE DESCRIPTOR GLOBAL 'Results' END-EXEC
```

Erläuterung

1. Der dynamische Cursor DYN_CURSOR wird deklariert. DYN_CURSOR besitzt eine dynamisch übersetzbare Cursorbeschreibung DYNSTMT.
2. Die Deskriptorbereiche „Placehold“ für die Platzhalter und „Results“ für die Spaltenwerte eines Ergebnissatzes werden angelegt.
3. Der Anweisungstext wird in SOURCESTMT eingelesen und für die spätere Ausführung vorbereitet. Der Anweisungstext kann eine dynamisch übersetzbare Anweisung oder eine Cursorbeschreibung sein.
4.
   a) Nach erfolgreicher Ausführung der PREPARE-Anweisung werden mit DESCRIBE INPUT Anzahl und Datentypen der Platzhalter in der dynamisch übersetzten SQL-Anweisung DYNSTMT abgefragt. Wenn das DESCRIBE INPUT erfolgreich war, dann ist DYNSTMT dynamisch übersetzt und der SQL-Deskriptorbereich „Placehold“ mit den Typbeschreibungen der dynamischen Parameter gesetzt.

b) Ebenso werden mit DESCRIBE OUTPUT Anzahl und Datentypen der Spalten eines Ergebnissatzes in den Deskriptorbereich „Results“ geschrieben. Wenn das DESCRIBE OUTPUT erfolgreich war, dann ist DYNSTMT dynamisch übersetzt und der SQL-Deskriptorbereich „Results“ mit den Typbeschreibungen der Spalten eines Ergebnissatzes gesetzt.

5. Mit GET DESCRIPTOR wird die Anzahl der Platzhalter H_COUNT bestimmt.
6. Für jeden Platzhalter werden mit GET DESCRIPTOR Datentyp, Länge, usw. gelesen und in die entsprechenden Benutzervariablen H_TYPE, H_LENGTH, usw. übertragen.
7. Entsprechend dem Datentyp des Platzhalters wird eine geeignete Benutzervariable ausgewählt. In diese wird der Wert des Platzhalters eingelesen. Um den Wert eines Platzhalters in den Deskriptorbereich zu übertragen, müssen mit SET DESCRIPTOR im Deskriptorbereich Datentyp, Länge, Anzahl der Dezimal- und Nachkommastellen, usw. den Datentypen der Benutzervariablen angepasst werden. Im Programm wurden z.B. drei Datentypen vorgesehen, die den SQL-Datentypen NUMERIC(18,0), NUMERIC(18,9) und NUMERIC(18,18) entsprechen. Je nach Datentyp des Platzhalters wird im Deskriptorbereich einer dieser Datentypen gesetzt.
   Nach der Ausführung der SET DESCRIPTOR-Anweisungen enthält der Deskriptorbereich „Placehold“ die Werte für die Platzhalter.
8. Mit GET DESCRIPTOR wird die Anzahl der Spalten der Ergebnistabelle abgefragt. Ist diese Anzahl 0, dann wird die dynamisch übersetzte Anweisung mit EXECUTE ausgeführt. In der USING-Klausel wird der Deskriptorbereich „Placehold“ angegeben. Er enthält die Werte für die Platzhalter in der dynamisch übersetzten Anweisung. Anschließend wird das Programm bei (13) fortgesetzt.
9. Ansonsten werden die Sätze der Ergebnistabelle mit Hilfe des bei (1) deklarierten Cursors gelesen.
   Analog (6) und (7) werden zunächst für den Deskriptorbereich „Results“ mit GET DESCRIPTOR Datentypen, Längen, usw. der Ergebnisspalten gelesen. Mit SET DESCRIPTOR werden dann Datentypen, Längen usw. im Deskriptorbereich angepasst.
10. Der Cursor DYN_CURSOR wird geöffnet. In der USING-Klausel wird der Deskriptorbereich „Placehold“ angegeben. Er enthält die Werte für die Platzhalter in der Cursorbeschreibung.
11. Die Ergebnistabelle wird satzweise mit FETCH in den Deskriptorbereich „Results“ geschrieben. Mit GET DESCRIPTOR werden die Spaltenwerte aus dem Deskriptorbereich in geeignete Benutzervariablen übertragen und ausgegeben.
12. Der Cursor DYN_CURSOR wird geschlossen.
13. Schließlich werden die Deskriptorbereiche mit DEALLOCATE wieder freigegeben.

5 Utility-Konzept

Dieses Kapitel beschreibt

- die Utility-Funktionen von SESAM/SQL
- die Systemeinbettung der Utility-Anweisungen
- die Programmeinbettung der Utility-Anweisungen.

Der Abschnitt „Utility-Funktionen von SESAM/SQL“ zeigt, welche Utility-Funktionen SESAM/SQL anbietet und durch welche Utility-Anweisungen diese Funktionen realisiert sind.

Der Abschnitt „Systemeinbettung der Utility-Anweisungen“ beschreibt Besonderheiten bei der Ausführung von Utility-Anweisungen durch SESAM/SQL. Außerdem behandelt der Abschnitt die unterschiedlichen Zustände von Anwender-Spaces, die sich durch die Ausführung von Utility-Anweisungen ergeben können.

Der Abschnitt „Programmeinbettung der Utility-Anweisungen“ geht kurz auf die Einbettung der Utility-Anweisungen in einem ESQL-Programm sowie auf Erfolgskontrolle und Fehlerbehandlung ein.

5.1 Utility-Funktionen von SESAM/SQL

Utility-Funktionen stellt SESAM/SQL über Utility-Anweisungen bereit.

Die folgende Tabelle zeigt, welche Utility-Funktionen SESAM/SQL anbietet und durch welche Utility-Anweisungen die einzelnen Utility-Funktionen realisiert werden.

Utility-Funktion	**Utility-Anweisung**
Eigenschaften der Datenbank ändern	ALTER CATALOG
Datenbank (Catalog-Space) anlegen	CREATE CATALOG
SESAM-Sicherungsbestand erstellen	COPY
Replikat erzeugen	CREATE REPLICATION
Replikat aktualisieren	REFRESH REPLICATION
Replikat erweitern	REFRESH SPACE
Datenbank, Catalog-Space, Anwender-Space(s) reparieren	RECOVER [USING]
Datenbank, Catalog-Space, Anwender-Space(s) zurücksetzen	RECOVER TO
Datenbank auf einen beliebigen Zeitpunkt zurücksetzen	RECOVER CATALOG [USING] TO ANY *zeitstempel*
Indizes neu aufbauen	RECOVER INDEX
Anwenderdaten in Basistabelle laden	LOAD
Anwenderdaten aus Basistabelle entladen	UNLOAD
Tabelle aus Datenbank in Export-Datei exportieren	EXPORT TABLE
Tabelle aus Export-Datei in Datenbank importieren	IMPORT TABLE
Mischen von Spaltenwerten, Anonymisieren von Daten	ALTER DATA FOR TABLE
Catalog-Space, Anwender-Spaces und Basistabellen reorganisieren	REORG
Partitionierung einer Basistabelle ändern	ALTER PARTITIONING FOR TABLE
Integritätsbedingungen prüfen	CHECK CONSTRAINTS
Basistabellen und Indizes formal prüfen	CHECK FORMAL
Medientabelle bearbeiten	ALTER MEDIA DESCRIPTION CREATE MEDIA DESCRIPTION DROP MEDIA DESCRIPTION
Informationen (Metadaten) über die SESAM-Sicherungsbestände pflegen	MODIFY
Datenbanken und Tabellen umstellen	MIGRATE

Tabelle 39: Utility-Funktionen von SESAM/SQL

Im Kapitel „Datenbank aufbauen, laden und warten“ (siehe Seite 339) sind die Utility-Anweisungen im Zusammenhang mit den Aufgaben beschrieben, die bei Aufbau und Wartung einer SESAM/SQL-Datenbank anfallen.

Vollständig beschrieben sind die einzelnen Utility-Anweisungen im Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“.

5.2 Systemeinbettung der Utility-Anweisungen

Utility-Anweisungen sind Anweisungen in SQL-Syntax, die jedoch nicht innerhalb von SQL-Transaktionen ausgeführt werden dürfen. Eine Utility-Anweisung eröffnet keine SQL-Transaktion und ist nicht rücksetzbar. Eine Utility-Anweisung eröffnet aber eine interne Transaktion, die mit Abschluss der Anweisung wieder beendet ist.
Utility-Anweisungen sind dynamisch übersetzbar (siehe Abschnitt „Dynamische SQL“ auf Seite 142).

5.2.1 Ausführung von Utility-Anweisungen durch SESAM/SQL

SESAM/SQL benutzt Sperrmechanismen für Transaktionen, um die Synchronisation von Utility-Anweisungen zu gewährleisten:

- Die Utility-Anweisung wartet das Ende aller offenen Transaktionen ab, die schon Sperren auf den betroffenen Spaces haben (siehe Seite 163).
- Während der Ausführung der Utility-Anweisung warten alle folgenden Zugriffe, wenn sie auf die betroffenen Spaces zugreifen wollen, bis die interne Transaktion der Utility-Anweisung beendet ist.

CALL-DML-Open werden geschlossen und gespeicherte SQL-Cursor werden entwertet, wenn die betroffene Tabelle von dem Utility geändert worden ist. Anschließende CALL-DML-Zugriffe erhalten den Status 9U, Zugriffe auf den SQL-Cursor den SQLSTATE 24SA5. Die entsprechenden Utility-Anweisungen sind:

– REORG [CATALOG_]SPACE
– CHECK CONSTRAINTS
– LOAD OFFLINE
– IMPORT TABLE
– RECOVER CATALOG / CATALOG_SPACE
– RECOVER SPACE
– REFRESH REPLICATION
– REFRESH SPACE
– ALTER CATALOG
– ALTER PARTITIONING FOR TABLE
– ALTER DATA FOR TABLE

Folgende Tabelle zeigt, welche Spaces von der Ausführung der einzelnen Utility-Anweisungen betroffen sind.

Utility-Anweisung	**von der Utility-Anweisung betroffene Spaces**
ALTER CATALOG	Catalog-Space
CREATE CATALOG	Catalog-Space
COPY	angegebene Sicherungseinheit (Catalog-Space und alle Anwender-Spaces, Catalog-Space, ein oder mehrere Anwender-Spaces)
CREATE REPLICATION	angegebene Einheit (Replikat)
REFRESH REPLICATION	angegebene Einheit (Replikat oder Teilreplikat)
REFRESH SPACE	angegebene Spaces
RECOVER [USING/TO]	angegebene Einheit für das Recovery (Catalog-Space und alle Anwender-Spaces, Catalog-Space, Space-Set, Space-Liste, einzelner Anwender-Space)
RECOVER USING/TO REPLICATION	angegebene Einheit für das Recovery (Catalog-Space und alle Anwender-Spaces des Replikats oder Teilreplikats, Catalog-Space, Space-Liste, einzelner Anwender-Space)
RECOVER INDEX	alle Anwender-Spaces, auf denen die angegebenen Indizes liegen
REORG [CATALOG_]SPACE	Catalog-Space, angegebener Anwender-Space
REORG ONLINE TABLE	Anwenderspace, auf dem die Basistabelle oder Partition der Basistabelle liegt
LOAD	Anwender-Space, auf dem die Basistabelle liegt, in die mit LOAD Anwenderdaten geladen werden
UNLOAD	Anwender-Space, auf dem die Basistabelle liegt, aus der mit UNLOAD Anwenderdaten entladen werden
EXPORT TABLE	Anwender-Space, auf dem die Basistabelle liegt, die in eine Export-Datei exportiert werden soll
IMPORT TABLE	Anwender-Space, in den mit Hilfe einer Export-Datei eine Basistabelle importiert werden soll; evtl. Anwender-Space, auf dem die Indizes aufgebaut werden sollen
ALTER DATA FOR TABLE	Anwender-Space, auf dem die Basistabelle liegt.
ALTER PARTITIONING FOR TABLE	Anwender-Spaces, auf denen die betroffenen Partitionen der Basistabelle liegen.
CHECK FORMAL	Anwender-Space der formal geprüft wird bzw. auf dem die zu prüfende Basistabelle oder der zu prüfende Index liegt

Tabelle 40: Von Utility-Anweisungen betroffene Spaces (Teil 1 von 2)

Utility-Anweisung	von der Utility-Anweisung betroffene Spaces
ALTER MEDIA DESCRIPTION CREATE MEDIA DESCRIPTION DROP MEDIA DESCRIPTION	Medientabelle auf dem Catalog-Space
MODIFY	Catalog-Tabellen RECOVERY_UNITS und DA_LOGS auf dem Catalog-Space (bzw. CAT-REC-Datei)
MIGRATE	Catalog-Space sowie Anwender-Space, auf dem die umzustellende Basistabelle liegt

Tabelle 40: Von Utility-Anweisungen betroffene Spaces (Teil 2 von 2)

Je nach Utility-Anweisung ist während der Ausführung der Anweisung ändernder oder lesender Zugriff anderer Benutzer auf einen von der Utility-Anweisung betroffenen Space möglich, eingeschränkt möglich oder überhaupt nicht möglich.

Paralleler Zugriff ist nicht erlaubt bei:

- ALTER CATALOG
- ALTER PARTITIONING FOR TABLE (nur auf die betroffenen Spaces)
- ALTER DATA FOR TABLE
- CREATE CATALOG
- CREATE REPLICATION
- MIGRATE
- RECOVER
- RECOVER INDEX
- REFRESH REPLICATION
- REFRESH SPACE
- REORG [CATALOG_]SPACE
  (während des Kopierens oder des Umbenennens der Arbeitsdatei in den Anwender-Space und während der Reorganisation des Catalog-Space)
- LOAD OFFLINE (falls die Klausel GENERATE INDEX angegeben ist)
- IMPORT TABLE

Parallel lesender Zugriff auf die Tabellen und Indizes des Space, die nicht von der Utility-Anweisung betroffen sind, ist erlaubt bei:

- CHECK CONSTRAINTS
- LOAD OFFLINE (falls die Klausel GENERATE INDEX nicht angegeben ist)

Parallel lesender Zugriff auf den gesamten von der Utility-Anweisung betroffenen Space ist erlaubt bei:

- CHECK FORMAL
- COPY
- UNLOAD
- EXPORT TABLE (wenn Anwenderdaten exportiert werden)
- REORG [CATALOG_]SPACE (für die Zeit des Aufbaus der Arbeitsdatei)

Parallel lesender und ändernder Zugriff auf den gesamten von der Utility-Anweisung betroffenen Space ist erlaubt bei:

- COPY ONLINE
- EXPORT TABLE (wenn keine Anwenderdaten exportiert werden)
- LOAD ONLINE
- MODIFY
- ALTER/CREATE/DROP MEDIA DESCRIPTION
- REORG ONLINE TABLE
- UNLOAD ONLINE

Nicht erlaubte Zugriffsversuche weist SESAM/SQL mit Statuscode ab und behandelt Spaces, Tabellen und Indizes, auf die kein Zugriff erlaubt ist, wie nicht verfügbare Spaces, Tabellen und Indizes.

Parallele RECOVER-Anweisungen

Eine Verkürzung von RECOVERY-Läufen kann erreicht werden, wenn der Anwender mehrere, parallel ablaufende RECOVER-Anweisungen gibt.

Parallel ablaufende RECOVER-Anweisungen dürfen sein:

- RECOVER eines einzelnen Space: `RECOVER SPACE` *space* `USING` *rec_unit*
- RECOVER einer Space-Liste: `RECOVER SPACE` *space*`,`*space*`[,...] USING` *rec_unit*
- RECOVER eines Space-Set: `RECOVER SPACESET AT CATALOG` *catalog* `USING` *zeitstempel*

Randbedingungen für die parallele Verarbeitung

- Die Space-Mengen der RECOVER-Anweisungen müssen disjunkt sein, d.h. ein Space darf nur von einer der parallelen RECOVER-Anweisungen betroffen sein.
- Für parallele RECOVER-Anweisungen dürfen die Parameter TO, RESTART und ADJUST nicht angegeben werden. Damit sind nur solche RECOVER-Anweisungen zulässig, bei denen Loggingdateien nachzufahren sind.
- Es müssen genügend Service-Tasks gestartet sein
  (ein Service-Task je paralleler RECOVER-Anweisung).

Ist eine dieser Randbedingungen nicht erfüllt, so werden die betroffenen RECOVER-Anweisungen durch die Sperrmechanismen serialisiert, siehe Seite 161.

Wenn auf den wiederherzustellenden Spaces eine partitionierte Tabelle liegt, dann sollten diese Spaces aus Konsistenzgründen in **einer** RECOVER-Anweisung wiederhergestellt werden.

5.2.2 Space-Zustände nach der Ausführung von Utility-Anweisungen

Die von der Ausführung einer Utility-Anweisung betroffenen Anwender-Spaces, Tabellen und Indizes bleiben in folgenden Fällen auch nach Ausführung der Utility-Anweisung gesperrt:

- Nach LOAD OFFLINE sind Indizes noch nicht neu aufgebaut.
  Zustand des Index: „defect“
  Zustand des Anwender-Space: O.K.
- Nach dem Laden mit LOAD OFFLINE in eine Basistabelle des Anwender-Space sind die Integritätsbedingungen noch nicht geprüft. Die Primärschlüsselbedingung wird bei der Ausführung von LOAD in jedem Fall geprüft.
  Zustand der Basistabelle: „check pending“
  Zustand des Anwender-Space: „check pending“
- Nach IMPORT TABLE in den folgenden Situationen:
  - Wenn Anwenderdaten importiert werden und für den Anwender-Space Logging eingeschaltet ist.
    Zustand des Anwender-Space: „copy pending“
  - Wenn Indizes noch nicht aufgebaut sind.
    Zustand des Index: „defect“
    Zustand des Anwender-Space: O.K.
- Nach ALTER DATA FOR TABLE muss vor der weiteren Bearbeitung des Anwender-Space ein SESAM-Sicherungsbestand erstellt werden, wenn für diesen Anwender-Space das Logging eingeschaltet ist.
  Zustand des Anwender-Space: „copy pending“
- Nach ALTER PARTITIONING FOR TABLE muss vor der weiteren Bearbeitung der Anwender-Spaces ein SESAM-Sicherungsbestand erstellt werden, wenn für diese Anwender-Spaces das Logging eingeschaltet ist.
  Zustand des Anwender-Space: „copy pending“
- Bei Ausführung von CHECK CONSTRAINTS hat SESAM/SQL die Verletzung von Integritätsbedingungen für mindestens eine Basistabelle des Anwender-Space festgestellt.
  Zustand der Basistabelle: „check pending“
  Zustand des Anwender-Space: „check pending“
- Nach RECOVER TO verletzen die Anwenderdaten Integritätsbedingungen.
  Zustand des Anwender-Space: „check pending“
- Nach LOAD OFFLINE und MIGRATE muss vor der weiteren Bearbeitung des Anwender-Space ein SESAM-Sicherungsbestand erstellt werden, wenn für diesen Anwender-Space das Logging eingeschaltet ist.
  Zustand des Anwender-Space: „copy pending“

- Nach MIGRATE, LOAD OFFLINE oder IMPORT TABLE, wenn das Zuladen aufgrund eines Fehlers abgebrochen wird.
  Zustand des Anwender-Space: „load running“
- Nach RECOVER USING, wenn CREATE INDEX oder DROP INDEX für eine partitionierte Tabelle auf diesem Space nachgefahren werden muss, aber nicht alle Anwender-Spaces, auf denen Partitionen dieser partitionierten Tabelle liegen, in der RECOVER-Anweisung enthalten sind.
  Zustand der Anwender-Spaces, auf denen Indizes liegen, die bei RECOVER neu aufgebaut wurden: „copy pending“
- Nach RECOVER TO in folgenden Situationen:
  - Wenn für den Anwender-Space Logging eingeschaltet ist und eine Anpassung der Daten an die Definitionen im Catalog-Space notwendig war (siehe Seite 240).
    Zustand des Anwender-Space: „copy pending“
  - Wenn Indizes neu aufgebaut werden mussten.
    Zustand der Anwender-Spaces, in denen Indizes liegen, die bei RECOVER TO neu aufgebaut wurden: „copy pending“
- Bei der Reparatur konnte der aktuelle Stand des Anwender-Space nicht wiederhergestellt werden (z.B. wegen fehlender oder defekter DA-LOG-Dateien).
  Zustand des Anwender-Space: „recover pending“
- Bei der Bearbeitung des Anwender-Space ist ein Fehler aufgetreten (z.B. fehlerhafte Beendigung einer Utility-Anweisung) oder SESAM/SQL stellt Datenfehler im Anwender-Space fest.
  Zustand des Anwender-Space: „space defect“

Wie der Datenbankverwalter gesperrte Anwender-Spaces bearbeiten kann, ist im Abschnitt „Gesperrte Anwender-Spaces bearbeiten“ auf Seite 365 beschrieben.

Welche Utility-Anweisungen in einem bestimmten Zustand eines Anwender-Space ausgeführt werden dürfen und in welchen neuen Zustand der Anwender-Space dadurch jeweils übergeführt wird, ist im Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“ beschrieben.

5.3 Programmeinbettung der Utility-Anweisungen

Programmstruktur

Utility-Anweisungen werden in gleicher Weise wie SQL-Anweisungen innerhalb eines ESQL-Programms angegeben. Allerdings dürfen Utility-Anweisungen nicht innerhalb von SQL-Transaktionen ausgeführt werden. Nähere Einzelheiten zur Programmstruktur sind im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“ beschrieben.

Erfolgskontrolle und Fehlerbehandlung

Die Rückmeldung auf eine Utility-Anweisung erfolgt in gleicher Weise wie bei SQL-Anweisungen: SESAM/SQL liefert einen SQL-Statuscode. Nähere Einzelheiten sind im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“ beschrieben.

6 Sicherheitskonzept

Sicherheit, d.h. der Schutz von Vertraulichkeit und Integrität der gespeicherten Informationen, ist ein wichtiger Aspekt heutiger DV- und SQL-Systeme.

Im Einzelnen werden folgende Sicherheitskriterien unterschieden:

- Zugangsschutz (Identifizierung, Authentisierung)
  Benutzer müssen vor einer Interaktion identifiziert und authentisiert werden.
- Zugriffsschutz (Rechteverwaltung, Rechteprüfung)
  Vom System sind Zugriffsrechte zwischen Benutzern (Subjekten) und Objekten zu verwalten. Bei jedem Zugriffsversuch von Benutzern auf Objekte, die der Rechteverwaltung unterliegen, muss das Betriebssystem die Berechtigung hierzu überprüfen. Unberechtigte Zugriffsversuche müssen abgewiesen werden.
- Beweissicherung
  Das System muss eine Protokollierungskomponente enthalten, die in der Lage ist, sicherheitsrelevante Ereignisse zu protokollieren.
- Wiederaufbereitung
  Sensitive Speicherobjekte müssen vor einer Wiederverwendung durch andere Benutzer so aufbereitet werden, dass keine Rückschlüsse auf ihren früheren Inhalt möglich sind.

Allgemeine Informationen zum Thema Sicherheit von DV-Systemen und Informationen zur Sicherheit in BS2000 finden Sie im Handbuch „SECOS (BS2000)“.

Der Sicherheitsbeauftragte des DV-Systems und der SESAM/SQL-Verwalter müssen anhand der Schutzmechanismen des DV-Systems, also von BS2000, und von SESAM/SQL ein Sicherkeitskonzept erarbeiten, das die gewünschte Sicherheit bietet und regelmäßige Prüfungen zulässt.
Die Schutzmechanismen von BS2000 sind im Handbuch „SECOS (BS2000)“, die Schutzmechanismen von SESAM/SQL in diesem Kapitel beschrieben.

Das Datenbanksystem SESAM/SQL bietet ein breites Spektrum von Schutzmechanismen für einen effektiven Datenschutz:

- Die Dateien einer SESAM/SQL-Datenbank können durch BS2000-Kennwörter geschützt werden, siehe Abschnitt „BS2000-Kennwörter für die Dateien der Datenbank“ auf Seite 171.

- Damit ein Benutzer auf einer SESAM/SQL-Datenbank arbeiten kann (Zugangsschutz), muss er die Berechtigung haben, auf diese Datenbank zuzugreifen, d.h., er muss dem Datenbanksystem als berechtigter Benutzer bekannt sein. Der Abschnitt „Zugangsberechtigung zu einer SESAM/SQL-Datenbank“ auf Seite 172 beschreibt die Voraussetzungen für das Arbeiten mit einer SESAM/SQL-Datenbank. Hier werden die Begriffe System-Benutzerkennung und Berechtigungsschlüssel eingeführt und das Einrichten eines Benutzers mit umfassenden Privilegien beschrieben.

  Der SESAM/SQL-Verwalter kann hier auch die technischen Möglichkeiten von BS2000 und von openUTM nutzen, z.B. die Zugangsprüfung über Chipkarte oder die elektronische Unterschrift.

- Der Benutzer muss das Privileg haben, die gewünschte Aktion in der Datenbank ausführen zu dürfen (Zugriffsschutz). Für jede Aktion, die der Benutzer ausführen möchte, muss ein entsprechendes Privileg vergeben worden sein. Der Abschnitt „Zugriffsschutz in SQL durch Privilegien“ auf Seite 176 beschreibt, wie der Zugriffsschutz in SQL über die Vergabe von Privilegien geregelt ist. Zunächst werden die Bestandteile des Zugriffsschutzkonzepts in SQL wie Sonder- und Tabellen-Privilegien erläutert. Außerdem wird beschrieben, was beim Vergeben und Entziehen von Privilegien zu beachten ist.

- Der Abschnitt „Zugriffsschutz in Verbindung mit dem Viewkonzept“ auf Seite 191 beschreibt, wie der Zugriffsschutz in SQL durch das Viewkonzept ergänzt wird.

- Auch für CALL-DML-Tabellen kann mit dem Dienstprogramm SEPA der Zugriffsschutz über Kennwörter realisiert werden, siehe Abschnitt „Kennwortschutz mit SEPA für CALL-DML-Tabellen“ auf Seite 192.

- Der Abschnitt „Zugriffsschutz durch Datenverschlüsselung“ auf Seite 193 beschreibt, wie sensitive Daten durch Verschlüsselung geschützt werden können. Diese Daten sind auch im Sinne des Sicherheitskriteriums „Wiederaufbereitung“ geschützt.
  Durch den Einsatz unterschiedlicher Schlüssel können Daten unterschiedlicher Sicherheitsbereiche (z.B. Kreditkarten-Nummern und Gesundheitsdaten) auch unterschiedlich verschlüsselt werden.

- Der Abschnitt „Schutz personenbezogener Daten durch Anonymisierung“ auf Seite 196 beschreibt die Anonymisierung sensitiver Daten mit SESAM/SQL.

- Der Abschnitt „Protokollierung sicherheitsrelevanter Ereignisse mit SAT“ auf Seite 198 beschreibt die Beweissicherungskomponente von SESAM/SQL.

- Die Dienstprogramme SESCOSP und SEDI70 (siehe Handbuch „Datenbankbetrieb“) protokollieren Datenbankzugriffe und erstellen Protokolle zur Auditierung des SESAM/SQL-Systems.

6.1 BS2000-Kennwörter für die Dateien der Datenbank

Die Dateien der SESAM/SQL-Datenbank (Datenbankdateien, datenbank-spezifische Dateien und DBH-spezifische Dateien) sind aus der Sicht des BS2000 normale Dateien, die wahlweise durch ein Kennwort geschützt werden können. Hierzu vergibt der Datenbankverwalter bereits beim Erstellen der Datenbank in der Utility-Anweisung CREATE CATALOG ein BS2000-Kennwort. Mit diesem Kennwort versieht SESAM/SQL dann automatisch alle Dateien der Datenbank.

Wenn die Datenbank auf einer DB-Kennung erstellt werden soll, muss der Datenbankverwalter entweder die DBH-Kennung in der DB-Kennung als Miteigentümer der betreffenden Dateien erklären oder die Dateien anlegen und dabei das BS2000-Kennwort selbst vergeben, siehe Abschnitt „Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen“ auf Seite 308. In der Utility-Anweisung CREATE CATALOG muss dieses Kennwort angegeben werden.

Um mit einer durch BS2000-Kennwörter geschützten SESAM/SQL-Datenbank arbeiten zu können, muss der Systemverwalter beim Starten des DBH das Kennwort in der DBH-Anweisung ADD-SQL-DATABASE-CATALOG-LIST angeben. Das Kennwort wird dann im SQL-Datenbankverzeichnis des DBH abgelegt.

Soll das Kennwort für eine SESAM/SQL-Datenbank geändert werden, so muss es der Datenbankverwalter für jede Datei der Datenbank gesondert mit dem Kommando MODIFY-FILE-ATTRIBUTES ändern. Dabei ist darauf zu achten, dass für jede Datei einer Datenbank dasselbe Kennwort zu vergeben ist.

Wurde das Kennwort für die Datenbank seit Erstellung eines bestimmten SESAM-Sicherungsbestands geändert, so muss beim Einspielen dieses SESAM-Sicherungsbestands mit der Utility-Anweisung RECOVER das Kennwort angegeben werden, das beim Erstellen des SESAM-Sicherungsbestands gültig war.

6.2 Zugangsberechtigung zu einer SESAM/SQL-Datenbank

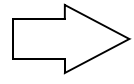
6.2.1 Systemzugang

Um überhaupt im BS2000 arbeiten zu können, benötigt ein Benutzer (TIAM- oder UTM-Anwender) eine BS2000-System-Benutzerkennung.
Ein TIAM-Anwender weist sich in BS2000 durch den symbolischen Rechnernamen und die BS2000-Benutzerkennung aus.
Ein UTM-Anwender weist sich im System openUTM aus durch den symbolischen Rechnernamen, den Namen der UTM-Anwendung und den KDCSIGN- bzw. LSES-Namen.

Um eine SESAM/SQL-Datenbank mit SQL bearbeiten zu können, benötigt ein BS2000-Benutzer einen Berechtigungsschlüssel eines SQL-Benutzers. Für diesen Berechtigungsschlüssel muss es einen entsprechenden Systemzugang in der Datenbank geben. Ein Systemzugang ist ein Paar, bestehend aus System-Benutzerkennung und Berechtigungsschlüssel. Ein SQL-Benutzer wird durch alle Systemzugänge im Datenbanksystem repräsentiert, die den Berechtigungsschlüssel dieses SQL-Benutzers enthalten.

Beim Umzug einer Datenbank auf einen anderen Rechner muss bereits **vor** dem Umzug ein Systemzugang für die Datenbank auf dem neuen Rechner erzeugt werden. Andernfalls besteht nach dem Umzug keine Möglichkeit mehr, auf die Datenbank zuzugreifen.

Systemzugang	
Berechtigungsschlüssel	**System-Benutzerkennung**
UNIVUSR	HOST2/BS2USERID2
AUTHORIZE1	HOST1/BS2USERID1
AUTHORIZE1	HOST1/UTM1/KDC1
AUTHORIZE1	HOST1/BS2USERID3
AUTHORIZE2	HOST2/UTM2/KDC2



SQL-Benutzer

Bild 9: Beispiel für Systemzugänge eines SQL-Benutzers

Wie sich ein SQL-Benutzer mit dem Berechtigungsschlüssel gegenüber SESAM/SQL ausweist, ist im Abschnitt „Berechtigungsschlüssel eines SQL-Benutzers bekannt geben“ auf Seite 174 beschrieben.

6.2.2 SQL-Benutzer mit universellen Befugnissen

Voraussetzung für den Aufbau einer SESAM/SQL-Datenbank ist, dass die System-Benutzerkennung einer befugten Person in der DBH-Option ADMINISTRATOR angegeben wird. Dieser Eintrag berechtigt den entsprechenden BS2000-Benutzer, die Utility-Anweisung CREATE CATALOG abzusetzen.

Beim Aufbauen einer Datenbank mit der Utility-Anweisung CREATE CATALOG wird u.a. ein Systemzugang erzeugt und im Catalog-Space der Datenbank hinterlegt: In der USER-Klausel bei CREATE CATALOG gibt der Datenbankverwalter einen Berechtigungsschlüssel an und ordnet ihn einer System-Benutzerkennung zu. Im Standardfall wird die System-Benutzerkennung mit der entsprechenden CREATE CATALOG-Berechtigung übernommen, d.h. die System-Benutzerkennung aus der DBH-Option ADMINISTRATOR.

Der mit CREATE CATALOG festgelegte Systemzugang ist der erste Systemzugang für die Datenbank und identifiziert den sog. universellen Benutzer. Der universelle Benutzer ist mit umfassenden Befugnissen zum Einrichten weiterer Benutzer und zur Vergabe von Privilegien ausgestattet. Somit kann der universelle Benutzer als die zentrale Instanz betrachtet werden, die alle Rechte insbesondere im Zusammenhang mit der Verwaltung der SESAM/SQL-Datenbank besitzt und diese Rechte beliebig unter anderen Benutzern aufteilen kann.

Zu diesem Zweck führt der universelle Benutzer folgende Maßnahmen durch:

1. Einrichten neuer SQL-Benutzer (siehe folgender Abschnitt)
2. Vergabe von Sonder-Privilegien an die neuen SQL-Benutzer (siehe Abschnitt „Zugriffsschutz in SQL durch Privilegien“ auf Seite 176)

6.2.3 Einrichten weiterer SQL-Benutzer durch den universellen Benutzer

Der universelle Benutzer richtet mit der SQL-Anweisung CREATE USER einen Berechtigungsschlüssel ein. Anschließend erstellt er mit CREATE SYSTEM_USER einen zugehörigen Systemzugang, indem er dem Berechtigungsschlüssel eine BS2000-System-Benutzerkennung zuordnet. Auf diese Weise eingerichtete Benutzer sind dem Datenbanksystem zwar bekannt, haben jedoch noch keinerlei Berechtigung, mit SQL auf Objekte der Datenbank zuzugreifen. Um auf Objekte der Datenbank zugreifen zu können, benötigt ein SQL-Benutzer entsprechende Privilegien (siehe Seite 176).

Der universelle Benutzer kann die Eigenschaft „universeller Benutzer“ an andere Benutzer weitergeben, indem er weitere Systemzugänge mit seinem Berechtigungsschlüssel vergibt. Alle System-Benutzerkennungen mit diesem Berechtigungsschlüssel sind dann aus Sicht des Datenbanksystems gleichberechtigt. Es versteht sich von selbst, dass diese Möglichkeit zur Vergabe universeller Berechtigungen nur äußerst restriktiv und wohlüberlegt eingesetzt werden sollte.

Der universelle Benutzer kann außerdem bereits vorhandenen Benutzern das Recht erteilen, weitere Benutzer einzurichten (siehe Abschnitt „Sonder-Privilegien“ auf Seite 176).

6.2.4 Berechtigungsschlüssel eines SQL-Benutzers bekannt geben

Ein SQL-Benutzer weist sich gegenüber SESAM/SQL mit dem Berechtigungsschlüssel aus. Den Berechtigungsschlüssel kann der Benutzer in der SOURCE-PROPERTIES-Option des ESQL-Precompilers unter AUTHORIZATION (siehe Handbuch „ESQL-COBOL für SESAM/SQL-Server“) angeben. Innerhalb eines ESQL-Programms kann mit der SQL-Anweisung SET SESSION AUTHORIZATION ein Berechtigungsschlüssel angegeben werden. Dieser Berechtigungsschlüssel wird nur dann aktueller Berechtigungsschlüssel, wenn in der SOURCE PROPERTIES-Option kein Berechtigungsschlüssel angegeben wurde. Für das Arbeiten mit dem Utility-Monitor muss der Berechtigungsschlüssel in der Konfigurationsdatei eingetragen sein.

6.2.5 Übersicht über Zugangsberechtigungen zu einer SESAM/SQL-Datenbank

wer / was	**wird festgelegt**		**durch Angabe von**
	wo	**mit**	
CREATE CATALOG-Berechtigter	DBH-Option	ADMINISTRATOR=...	neuer System-Benutzer-kennung
Universeller Benutzer	ESQL-Programm Utility-Monitor	CREATE CATALOG USER=... PASSWORD=...	erstem Systemzugang
BS2000-Kennwort für die Datenbank			BS2000-Kennwort
SQL-Benutzer		CREATE USER CREATE SYSTEM_USER	neuem Berechtigungs-schlüssel neuem Systemzugang: (System-Benutzerken-nung, Berechtigungs-schlüssel)

Tabelle 41: Einrichten neuer SQL-Benutzer in SESAM/SQL und Vergabe des BS2000-Kennworts

wer / was	**wird bekanntgegeben**		**durch Angabe von**
	wo	**mit**	
Berechtigungs-schlüssel für SQL-Benutzer	Precompiler-Option	SOURCE-PROPERTIES-AUTHORIZATION=...	Berechtigungsschlüssel
	Konfigurationsdatei bzw. entsprechen-de Maske des Utility-Monitors	SEE-AUTHID=... Parameter für Utility-Monitor	Berechtigungsschlüssel
	ESQL-Programm Utility-Monitor	SET SESSION AUTHORIZATION	Berechtigungsschlüssel
BS2000-Kennwort für die Datenbank	DBH-Startanwei-sung	ADD-SQL-DATABA-SE-CATALOG-LIST ... PASSWORD=...	BS2000-Kennwort

Tabelle 42: Bekannt geben von SQL-Benutzern in SESAM/SQL und des BS2000-Kennworts

6.3 Zugriffsschutz in SQL durch Privilegien

Der Zugriffsschutz wird in SQL über die Vergabe von Privilegien geregelt. Durch Vergabe von Privilegien berechtigt ein autorisierter Benutzer, der sog. Grantor, einen anderen Benutzer, den sog. Grantee, eine bestimmte Aktion auf einem bestimmten Objekt auszuführen.

SESAM/SQL unterscheidet Sonder-Privilegien, Tabellen-Privilegien und Privilegien für Routinen. Bei Sonder-Privilegien ist das Objekt die Datenbank oder, beim Privileg USAGE ON STOGROUP, eine Storage Group. Bei Tabellen-Privilegien ist das Objekt eine bestimmte Tabelle innerhalb eines bestimmten Schemas. Bei Privilegien für Routinen ist das Objekt eine Prozedur oder eine User Defined Function (UDF) innerhalb eines Schemas.

Ein Privileg wird somit identifiziert durch Grantor, Grantee, Aktion und Objekt. Demnach sind zwei ansonsten identische Privilegien, die demselben Grantee von unterschiedlichen Grantoren erteilt wurden, als voneinander verschieden anzusehen.

6.3.1 Sonder-Privilegien

Die Berechtigung zur Vergabe von Sonder-Privilegien mit Ausnahme von USAGE ON STOGROUP hat der universelle Benutzer. Die Berechtigung zur Vergabe des Sonderprivilegs USAGE ON STOGROUP für eine bestimmte Storage Group hat der Eigentümer dieser Storage Group.

Einrichten und Löschen von SQL-Benutzern (CREATE USER)

Das Sonder-Privileg CREATE USER berechtigt den Grantee, folgende Aufgaben wahrzunehmen:

- Einrichten eines neuen SQL-Benutzers
  Ein SQL-Benutzer wird eingerichtet, indem mit der SQL-Anweisung CREATE USER zunächst ein Berechtigungsschlüssel für diesen Benutzer vergeben wird.
  Anschließend muss diesem Berechtigungsschlüssel mit der SQL-Anweisung CREATE SYSTEM_USER eine System-Benutzerkennung zugeordnet werden. Auf diese Weise können für einen SQL-Benutzer ein oder mehrere Systemzugänge erstellt werden, die diesen SQL-Benutzer repräsentieren.
- Löschen eines SQL-Benutzers
  Ein Benutzer wird gelöscht, indem mit der SQL-Anweisung DROP USER alle Systemzugänge gelöscht werden, die den Berechtigungsschlüssel dieses SQL-Benutzers enthalten.
- Entfernen eines bestimmten Systemzugangs für einen bestimmten Benutzer mit der SQL-Anweisung DROP SYSTEM_USER.

Erstellen eines Schemas (CREATE SCHEMA)

Das Sonder-Privileg CREATE SCHEMA berechtigt einen Benutzer, ein Schema zu erstellen. Der Eigentümer dieses Schemas hat dann für dieses Schema alle Tabellen-Privilegien (siehe Abschnitt „Tabellen-Privilegien“ unten). Eigentümer eines Schemas ist der Benutzer, der die zugehörige CREATE SCHEMA-Anweisung eingegeben hat bzw. der in der AUTHORIZATION-Klausel von CREATE SCHEMA angegebene Benutzer.

Ausführen der Utilities (UTILITY)

Das Sonder-Privileg UTILITY berechtigt einen Benutzer, Utilities auf der Datenbank auszuführen.

Festlegen der Speichermedien für die Spaces (CREATE STOGROUP)

Das Sonder-Privileg CREATE STOGROUP berechtigt einen Benutzer, die SQL-Anweisung CREATE STOGROUP abzusetzen und somit die Speichermedien anzugeben, auf denen später Anwender-Spaces angelegt werden sollen. Dieser Benutzer wird der Eigentümer der betreffenden Storage Group und erhält für diese Storage Group das Sonder-Privileg USAGE ON STOGROUP.

Verwenden einer Storage Group (USAGE ON STOGROUP)

Der Eigentümer einer Storage Group hat das Sonder-Privileg USAGE ON STOGROUP für diese Storage Group. Das Sonder-Privileg USAGE ON STOGROUP berechtigt einen Benutzer, einen Space mit CREATE SPACE auf einer bestimmten Storage Group anzulegen. Will ein Benutzer einen Anwender-Space mit ALTER SPACE ändern, muss der Benutzer das Sonder-Privileg USAGE ON STOGROUP besitzen, wenn bei ALTER SPACE die USING STOGROUP-Klausel angegeben ist.

Alle Sonder-Privilegien (ALL SPECIAL PRIVILEGES)

Über ALL SPECIAL PRIVILEGES erhält der Grantee alle Sonder-Privilegien außer USAGE-STOGROUP, die der Grantor für eine bestimmte Datenbank bzw. eine bestimmte Storage Group weitervergeben darf.

6.3.2 Tabellen-Privilegien

Ein Benutzer, der mit CREATE SCHEMA (ohne AUTHORIZATION-Klausel) ein Schema erstellt, bzw. ein in der AUTHORIZATION-Klausel angegebener Benutzer, ist der Eigentümer dieses Schemas. Als Eigentümer eines Schemas besitzt er alle im Folgenden beschriebenen Tabellen-Privilegien für Objekte seines Schemas. Außerdem ist er berechtigt, Tabellen-Privilegien seines Schemas als Grantor an andere Benutzer weiterzugeben.

Auswählen von Sätzen (SELECT)

Das Tabellen-Privileg SELECT für eine bestimmte Tabelle berechtigt einen Benutzer, Sätze in dieser Tabelle auszuwählen sowie mit CREATE VIEW einen View für diese Tabelle zu definieren. Das Tabellen-Privileg SELECT bleibt gültig, wenn nach der Vergabe des Privilegs Spalten hinzugefügt wurden.

Einfügen von Sätzen (INSERT)

Das Tabellen-Privileg INSERT für eine bestimmte Tabelle berechtigt einen Benutzer, Sätze in diese Tabelle einzufügen. Voraussetzung für das Einfügen von Sätzen ist, dass der Benutzer das Tabellen-Privileg SELECT für die im Abfrage-Ausdruck der INSERT- oder MERGE-Anweisung genannten Tabellen besitzt. Das Tabellen-Privileg INSERT bleibt gültig, wenn nach der Vergabe des Privilegs Spalten hinzugefügt wurden.

Löschen von Sätzen (DELETE)

Das Tabellen-Privileg DELETE für eine bestimmte Tabelle berechtigt einen Benutzer, Sätze einer bestimmten Tabelle zu löschen. Voraussetzung für das Löschen von Sätzen ist, dass der Benutzer das Tabellen-Privileg SELECT für die in der Suchbedingung der DELETE-Anweisung genannten Tabellen besitzt. Das Tabellen-Privileg DELETE bleibt gültig, wenn nach der Vergabe des Privilegs Spalten hinzugefügt wurden.

Ändern von Spaltenwerten (UPDATE)

Das Tabellen-Privileg UPDATE für ausgewählte Spalten einer bestimmten Tabelle berechtigt einen Benutzer, in Sätzen dieser Tabelle Werte der angegebenen Spalten zu ändern. Voraussetzung für das Ändern von Spaltenwerten ist, dass der Benutzer das Tabellen-Privileg SELECT für die in der Suchbedingung der UPDATE- oder MERGE-Anweisung genannten Tabellen besitzt. Werden keine Spalten beim Tabellen-Privileg UPDATE angegeben, gilt es für die gesamte Tabelle, einschließlich aller nach der Vergabe des Privilegs hinzugefügten Spalten.

Angeben von Spalten in Integritätsbedingungen (REFERENCES)

Das Tabellen-Privileg REFERENCES für ausgewählte Spalten einer bestimmten Tabelle berechtigt einen Benutzer, diese Spalten bei der Definition von Integritätsbedingungen zu verwenden. Werden keine Spalten beim Tabellen-Privileg angegeben, gilt es für die gesamte Tabelle, einschließlich aller nach der Vergabe des Privilegs hinzugefügten Spalten.

Alle Privilegien (ALL PRIVILEGES)

Über ALL PRIVILEGES für eine bestimmte Tabelle erhält der Grantee alle Privilegien, die der Grantor für diese Tabelle weitergeben darf.

6.3.3 Privilegien für Routinen

Der Berechtigungsschlüssel, der eine Routine (Prozedur oder eine User Defined Function (UDF)) ausführen möchte, benötigt das EXECUTE-Privileg für diese Routine.

Zur Ausführung einer Routine ist es nicht erforderlich, die einzelnen Tabellen- oder Spalten-Privilegien zu besitzen, die für die Ausführung der in der Routine enthaltenen DML-Anweisungen nötig sind.

Der Berechtigungsschlüssel, der die Routine erzeugt, erhält automatisch das EXECUTE-Privileg für diese Routine.

Der Berechtigungsschlüssel muss das EXECUTE-Privileg für die in der Routine direkt aufgerufenen Routinen besitzen. Zusätzlich muss er für alle Tabellen und Spalten, die in der Routine angesprochen werden, diejenigen Privilegien besitzen, die benötigt werden, um die in der Routine enthaltenen DML-Anweisungen ausführen zu können.

Hat er für die betreffenden Privilegien sogar die Berechtigung, diese weitergeben zu dürfen, dann darf er auch das EXECUTE-Privileg an andere Berechtigungsschlüssel weitergeben und wieder entziehen.

6.3.4 Privilegien vergeben mit der SQL-Anweisung GRANT

Die folgenden Benutzer besitzen bestimmte Privilegien:

- Der universelle Benutzer besitzt alle Sonder-Privilegien für eine Datenbank mit Ausnahme von USAGE ON STOGROUP.
- Der Eigentümer einer Storage Group besitzt das Sonder-Privileg USAGE ON STOGROUP für diese Storage Group.
- Der Eigentümer eines Schemas besitzt alle Tabellen-Privilegien für die Basistabellen seines Schemas.
- Welche Zugriffsrechte der Eigentümer eines View für den View besitzt, ist auf Seite 181 beschrieben.

Universeller Benutzer, alle Schema-Eigentümer und Eigentümer von Storage Groups sowie Eigentümer von Views, die ihre Privilegien mit „WITH GRANT OPTION“ erhalten haben, können ihre jeweiligen Privilegien mit der SQL-Anweisung GRANT an andere Benutzer weitergeben.

Die SQL-Anweisung GRANT lässt sich schematisch wie folgt darstellen:

```
GRANT Privileg(ien) ON Objekt TO Grantee(s) [WITH GRANT OPTION]
```

GRANT kann als eigenständige SQL-Anweisung oder als Teil der CREATE SCHEMA-Anweisung angegeben werden.

„Grantor“ ist der Benutzer, der das Privileg mit GRANT erteilt, „Grantee“ ist der Benutzer, dem dieses Privileg erteilt wird. „Objekt“ kann je nach Privileg eine Tabelle oder eine Datenbank bzw. eine Storage Group sein.
Ein Benutzer kann ein ihm mit der GRANT-Anweisung erteiltes Privileg nur dann als Grantor an andere Benutzer weitergeben, wenn es ihm „WITH GRANT OPTION“ erteilt wurde. Der Grantee kann also seinerseits zum Grantor werden und das Privileg wiederum mit oder ohne Zusatz „WITH GRANT OPTION“ vergeben. Auf diese Weise lässt sich ein hierarchisch organisierter Zugriffsschutz realisieren.

Für „Grantee“ kann der Grantor in der GRANT-Anweisung die Berechtigungsschlüssel ausgewählter Benutzer oder das Schlüsselwort PUBLIC angeben. Mit PUBLIC wird das Privileg allen Benutzern erteilt, die berechtigt sind, mit der Datenbank zu arbeiten.

Vergibt ein Grantor einem Grantee dasselbe Privileg einmal mit und einmal ohne die Klausel „WITH GRANT OPTION“, dann ist stets „WITH GRANT OPTION“ wirksam.

Mit der GRANT-Anweisung kann ein Grantor ein Tabellen-Privileg auch an den Eigentümer eines anderen Schemas vergeben. Auf diese Weise kann ein Schema-Eigentümer z.B. Views für sein Schema definieren, denen Tabellen eines anderen Eigentümers zu Grunde liegen.

Sonder-Privilegien und Tabellen-Privilegien sollten nach einem wohlüberlegten Datenschutzkonzept vergeben werden. Insbesondere sollten Sonder-Privilegien nur einem ausgewählten Benutzerkreis zur Verfügung gestellt werden, wobei die Klausel „WITH GRANT OPTION“ sehr restriktiv zu verwenden ist.

Besonderheiten bei der Privilegien-Vergabe für das Erstellen von Views

Die Berechtigung, mit CREATE VIEW einen View zu erstellen, erhält ein Benutzer nur über das Tabellen-Privileg SELECT für die diesem View zu Grunde liegenden Tabellen. Zwei Fälle sind zu unterscheiden:

- Der View ist änderbar.
  In diesem Fall liegt dem View **eine** Basistabelle zu Grunde. Besitzt der Eigentümer des View für die zu Grunde liegende Basistabelle die Tabellen-Privilegien INSERT, UPDATE oder DELETE, dann erhält der Eigentümer auch automatisch das entsprechende Tabellen-Privileg für den View.
- Der View ist nicht änderbar.
  Der Eigentümer erhält automatisch das Tabellen-Privileg SELECT für den View.

Die automatische Vergabe der Privilegien für den View kann man sich so vorstellen, dass SESAM/SQL mit einer impliziten GRANT-Anweisung ein entsprechendes Tabellen-Privileg für den View erteilt.

Der View-Eigentümer kann ein Tabellenprivileg für einen View nur dann mit GRANT weitergeben, wenn er es für die dem View zu Grunde liegenden Basistabellen „WITH GRANT OPTION“ besitzt.

Beispiel

Das folgende Beispiel skizziert die Privilegienvergabe für den Eigentümer eines View und die Weitergabe von Privilegien an einen anderen Benutzer.

Benutzer A ist der Eigentümer der Basistabelle T und besitzt damit auch die Tabellen-Privilegien SELECT, INSERT, DELETE, UPDATE und REFERENCES. A ist berechtigt, diese Privilegien an andere Benutzer weiterzugeben.

A erteilt nun das Tabellen-Privileg DELETE „WITH GRANT OPTION“ und das Tabellen-Privileg SELECT an den Benutzer B:

```
GRANT DELETE ON T TO B WITH GRANT OPTION
GRANT SELECT ON T TO B
```

Aufgrund der SELECT-Berechtigung für die Basistabelle T darf B einen View auf T erzeugen:

```
CREATE VIEW V AS SELECT * FROM T
```

B erhält damit automatisch, d.h. von SESAM/SQL, die für die Tabelle T erteilten Tabellen-Privilegien (SELECT und DELETE „WITH GRANT OPTION“) für den View V. Damit bestehen für B folgende Tabellen-Privilegien für T und V:

Grantor	Grantee	Objekt	Tabellen-Privileg	WITH GRANT OPTION
A	B	T	DELETE	ja
A	B	T	SELECT	nein
SESAM/SQL	B	V	DELETE	ja
SESAM/SQL	B	V	SELECT	nein

Tabelle 43: Tabellen-Privilegien für Basistabelle T und View V

Anschließend werden von A mit GRANT folgende Tabellen-Privilegien erteilt:

```
GRANT UPDATE ON T TO B,C
GRANT INSERT ON T TO B WITH GRANT OPTION
```

B erteilt das folgende Tabellen-Privileg:

```
GRANT DELETE ON V TO D WITH GRANT OPTION
```

Damit sind folgende Tabellen-Privilegien neu hinzugekommen:

Grantor	Grantee	Objekt	Tabellen-Privileg	WITH GRANT OPTION
A	B	T	UPDATE	nein
A	C	T	UPDATE	nein
SESAM/SQL	B	V	UPDATE	nein
A	B	T	INSERT	ja
SESAM/SQL	B	V	INSERT	ja
B	D	V	DELETE	ja

Tabelle 44: Neu hinzugekommene Tabellen-Privilegien für Basistabelle T und View V

6.3.5 Privilegien entziehen mit der SQL-Anweisung REVOKE

Mit der SQL-Anweisung REVOKE kann ein Benutzer Privilegien wieder entziehen, die er als Grantor erteilt hat. Um konsistenten Zugriffschutz zu gewährleisten, verlangt SQL die Einhaltung bestimmter Regeln bei der Rücknahme von Privilegien.

Die SQL-Anweisung REVOKE lässt sich schematisch wie folgt darstellen:

```
REVOKE Privileg(ien) ON Objekt FROM Grantee(s) { CASCADE | RESTRICT }
```

Mit REVOKE entzieht ein Benutzer den „Grantees“ die angegebenen „Privilegien“. „Objekt“ kann je nach Privileg eine Tabelle oder eine Datenbank bzw. eine Storage Group sein. Ein Privileg darf einem Grantee nur von dem Benutzer entzogen werden, der ihm dieses Privileg zuvor als Grantor erteilt hat.

Mit REVOKE ... FROM PUBLIC kann ein Benutzer ein bestimmtes Privileg allen anderen Benutzern entziehen, wenn er es zuvor als Grantor mit GRANT... TO PUBLIC erteilt hat.

Mit REVOKE ... RESTRICT kann ein Benutzer die angegebenen Privilegien den Grantees nur entziehen, wenn die Grantees die Privilegien nicht an andere Benutzer weitergegeben haben oder wenn diese anderen Benutzer die Privilegien nicht mehr besitzen. Gibt es noch solche weitergegebenen Privilegien, dann werden die mit REVOKE angegebenen Privilegien nicht entzogen und es erfolgt eine Fehlermeldung.

Mit REVOKE ... CASCADE dagegen kann ein Benutzer den Grantees die angegebenen Privilegien in jedem Fall entziehen. Dabei werden außerdem alle Privilegien entzogen, die auf Grund der angegebenen Privilegien von den Grantees an andere Benutzer weitergegeben wurden.

Der Benutzer kann mit Hilfe der Tabellen des INFORMATION_SCHEMA ermitteln, in welcher Reihenfolge die Privilegien mit REVOKE ... RESTRICT entzogen werden müssen. Das INFORMATION_SCHEMA beschreibt in den Tabellen TABLE_PRIVILEGES, COLUMN_PRIVILEGES, USAGE_PRIVILEGES und CATALOG_PRIVILEGES, welche Privilegien welchen Berechtigungsschlüsseln zugeordnet sind.

Da mit **einer** REVOKE ... CASCADE-Anweisung unter Umständen sehr viele Privilegien entzogen und auch Views (siehe Seite 190) und Referenzbedingungen (siehe Seite 189) gelöscht werden können, empfiehlt es sich, sich auch vor einem REVOKE ... CASCADE einen Überblick über bestehende Privilegien in den Tabellen des INFORMATION_SCHEMA zu verschaffen.

Wie bereits erwähnt, werden zwei ansonsten identische Privilegien, die einem Grantee C von zwei verschiedenen Grantoren A und B erteilt wurden, als voneinander verschieden angesehen.

A hat z.B. dasselbe Privileg einmal an C und einmal „WITH GRANT OPTION“ an B weitergegeben; B seinerseits gibt dieses Privileg ebenfalls an C weiter. Entzieht nun A dem Grantee C dieses Privileg mit `REVOKE` *Privileg* `ON` *Objekt* `FROM C RESTRICT`, dann besitzt C immer noch das von B erteilte Privileg. Erst wenn auch B dieses Privileg dem Grantee C entziehen würde, ist es C entzogen.

Entzieht stattdessen nun A zusätzlich dem Grantee B mit `REVOKE` *Privileg* `ON` *Objekt* `FROM B CASCADE` das Privileg, dann besitzen B und C das Privileg nicht mehr.

Im Folgenden wird das Entziehen von Privilegien mit REVOKE anhand eines einfachen Beispiels erläutert.

Ein Privileg kann allgemein identifiziert werden durch
(Grantor, Grantee, Aktion, Objekt, [WITH GRANT OPTION]).

Es liegt folgende Ausgangssituation vor:

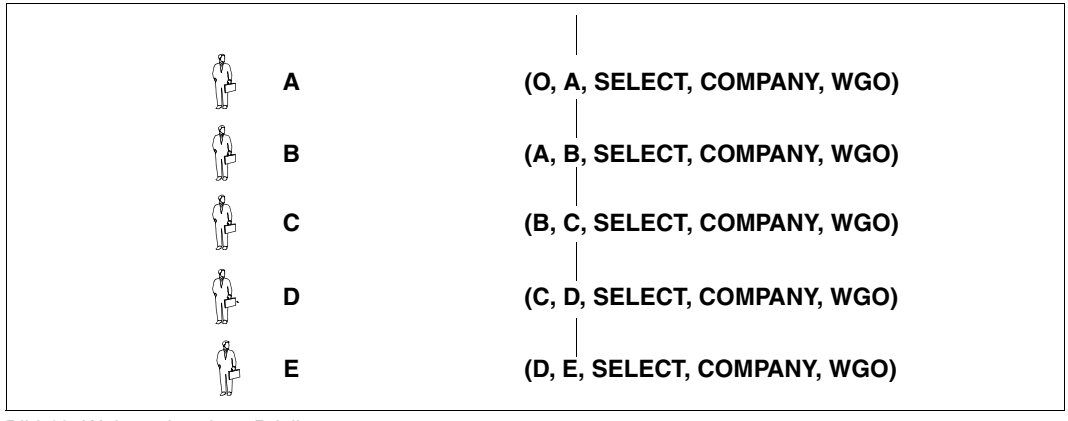


Bild 10: Weitergabe eines Privilegs

Der Eigentümer O des Schemas hat dem Benutzer A das Tabellen-Privileg SELECT auf der Tabelle COMPANY mit der Klausel „WITH GRANT OPTION“ (im Beispiel als WGO abgekürzt) erteilt. Dieses Privileg ist dann beschrieben durch (O, A, SELECT, COMPANY, WGO), siehe Bild 10:

Dieses Privileg hat Benutzer A an Benutzer B „WITH GRANT OPTION“ weitergegeben. B wiederum hat das Privileg an C weitergegeben, C an D und D schließlich an E. B, C und D haben das Privileg jeweils „WITH GRANT OPTION“ erteilt.
Somit bestehen folgende Privilegien:
(O, A, SELECT, COMPANY, WGO)
(A, B, SELECT, COMPANY, WGO)
(B, C, SELECT, COMPANY, WGO)
(C, D, SELECT, COMPANY, WGO)
(D, E, SELECT, COMPANY, WGO)

In Bild 10 wird die Tatsache „Benutzer B hat das Privileg von Benutzer A erhalten“ dadurch verdeutlicht, dass B direkt unter A liegt. Eine Linie zwischen zugehörigen Privilegien drückt aus, dass A das Privileg B noch nicht entzogen hat.

D entzieht nun E das Privileg mit der Anweisung:

```
REVOKE SELECT ON company FROM E RESTRICT
```

Nachdem der REVOKE erfolgreich durchgeführt wurde, besitzt E nicht mehr das Privileg (D, E, SELECT, COMPANY, WGO), siehe Bild 11.

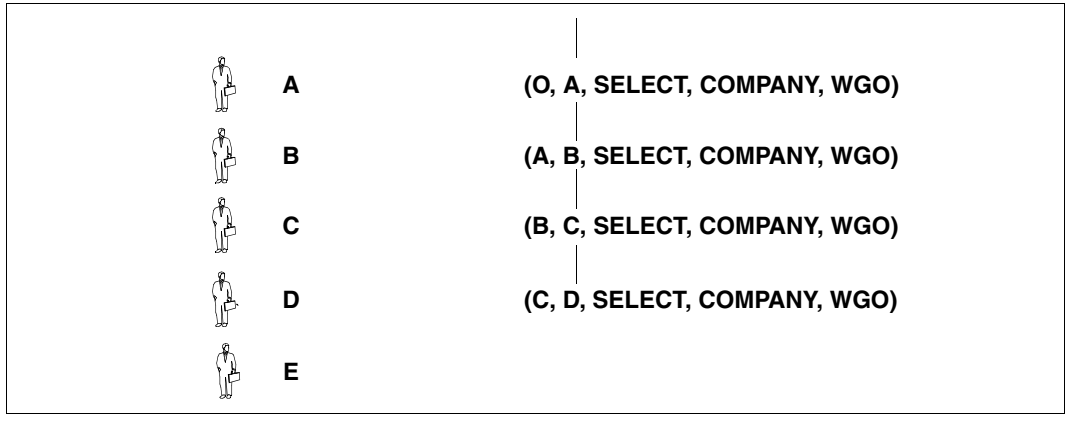


Bild 11: Entzug des Privilegs (D,E,SELECT,COMPANY,WGO)

Analog könnte nun D das Privileg durch C entzogen werden, anschließend könnte C das Privileg durch B entzogen werden, usw.. Voraussetzung dafür, dass D dem Benutzer E das Privileg erfolgreich entziehen kann, ist, dass nicht weitere Benutzer dasselbe Privileg von E erhalten haben und noch besitzen. In Bild 10 und Bild 11 drückt sich dies wie folgt aus: Von E bzw. D führen keine Linien zu einem Benutzer unterhalb von E bzw. D.

Vermeidung abgetrennter Privilegien

In der im Bild 11 dargestellten Situation möchte A nun B das Privileg entziehen. Der folgende unzulässige REVOKE wird jedoch abgewiesen:

```
REVOKE SELECT ON company FROM B RESTRICT
```

Die Ausführung des REVOKE hätte nämlich zu folgender Situation geführt:

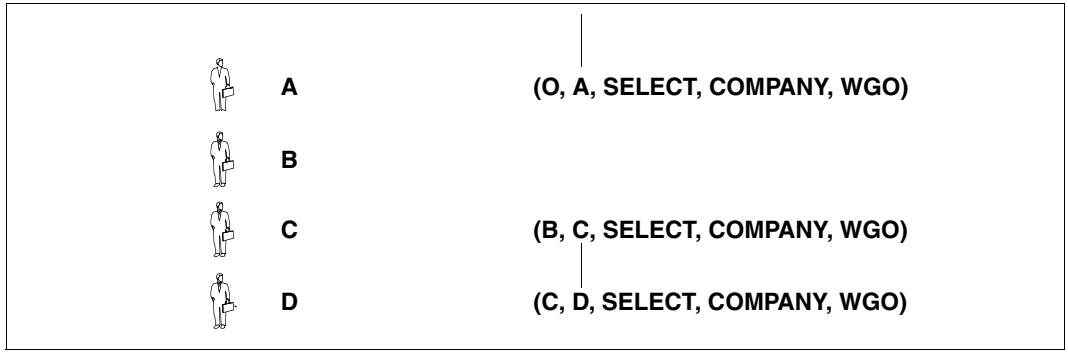


Bild 12: Hypothetische Situation: abgetrennte Privilegien

C hat weiterhin das von B erteilte Privileg (B, C, SELECT, COMPANY, WGO). B selbst besitzt das ihm von A erteilte Privileg (A, B, SELECT, COMPANY, WGO) nicht mehr, d.h. das Privileg (B, C, SELECT, COMPANY, WGO) in Bild 12 ist gleichsam „abgetrennt“. Damit ist aber auch das auf Grund des abgetrennten Privilegs (B, C, SELECT, COMPANY, WGO) erteilte Privileg (C, D, SELECT, COMPANY, WGO) abgetrennt.

In Bild 12 sind abgetrennte Privilegien so gekennzeichnet:
Es gibt von einem solchen Privileg keine Verbindungslinie mehr nach oben zu einem Privileg oder es gibt eine Verbindungslinie nach oben zu einem abgetrennten Privileg.

Allgemein heißt ein erteiltes und noch bestehendes Privileg abgetrennt (englisch „abandoned“), wenn das Privileg noch existiert, aber der Grantor dieses Privileg nicht mehr besitzt. Jedes Privileg, das sich aus einem abgetrennten Privileg ableitet, heißt ebenfalls abgetrennt. Der Begriff des abgetrennten Privilegs bezeichnet eine rein hypothetische Situation, da SESAM/SQL das Entstehen von abgetrennten Privilegien bei REVOKE verhindert.

Bei REVOKE ... RESTRICT verhindert SESAM/SQL das Entstehen von abgetrennten Privilegien, indem es einen REVOKE ... RESTRICT abweist, dessen Ausführung zu einem abgetrennten Privileg führen würde. Vor jedem REVOKE ... RESTRICT sollte sich der Benutzer daher vergewissern, ob ein Entziehen eines bestimmten Privilegs zu abgetrennten Privilegien führen und deshalb abgewiesen würde. Solche Privilegien müssen zuvor mit REVOKE ... RESTRICT durch den Benutzer entzogen werden, der dieses Privileg als Grantor erteilt hat.

Die Ausführung eines REVOKE ... CASCADE kann dagegen nie zu einer Situation führen, in der abgetrennte Privilegien entstehen können. Deswegen wird ein REVOKE ... CASCADE auch nie aus diesem Grund abgewiesen. Entzieht nämlich z.B. in der in Bild 10 dargestellten Situation Grantor A dem Grantee B mit `REVOKE SELECT ON company FROM B CASCADE` das SELECT-Privileg, dann werden automatisch in einer Art „Kettenreaktion“ ebenfalls alle Privilegien entzogen, die auf Grund des an B „WITH GRANT OPTION“ vergebenen Privilegs von anderen Grantees vergeben wurden: Zunächst entzieht Grantor D dem Grantee E das Privileg (D, E, SELECT, COMPANY, WGO), dann Grantor C dem Grantee D das Privileg (C, D, SELECT, COMPANY, WGO), usw. Auf diese Weise kann es nie zu abgetrennten Privilegien kommen.

Die beiden folgenden Beispiele beschreiben Situationen, in denen abgetrennte Privilegien entstehen bzw. nicht entstehen würden.

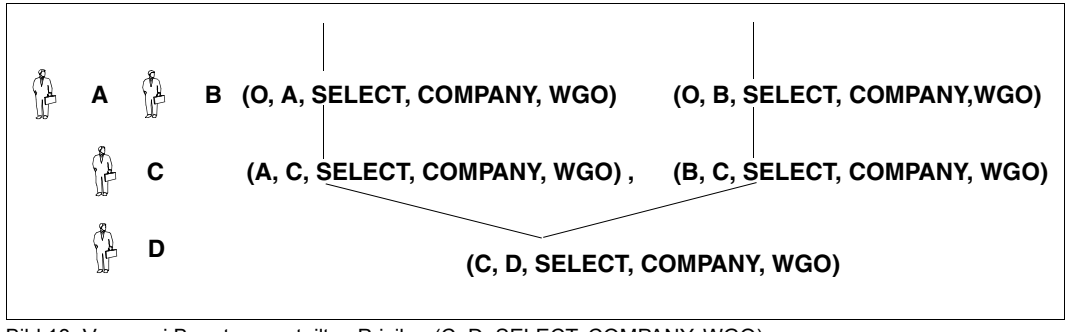


Bild 13: Von zwei Benutzern erteiltes Privileg (C, D, SELECT, COMPANY, WGO)

Bild 13 veranschaulicht folgende Ausgangssituation: C hat von zwei verschiedenen Benutzern (Grantor A und Grantor B) das ansonsten gleiche Privileg erhalten. C hat seinerseits dieses Privileg an D weitergegeben. Nun möchte A dem Benutzer C das zuvor erteilte Privileg (A, C, SELECT, COMPANY, WGO) entziehen.

Zu diesem Zweck gibt A die Anweisung: `REVOKE SELECT ON company FROM C RESTRICT`

Der REVOKE ist erfolgreich. Bild 14 zeigt die Situation nach ausgeführtem REVOKE.

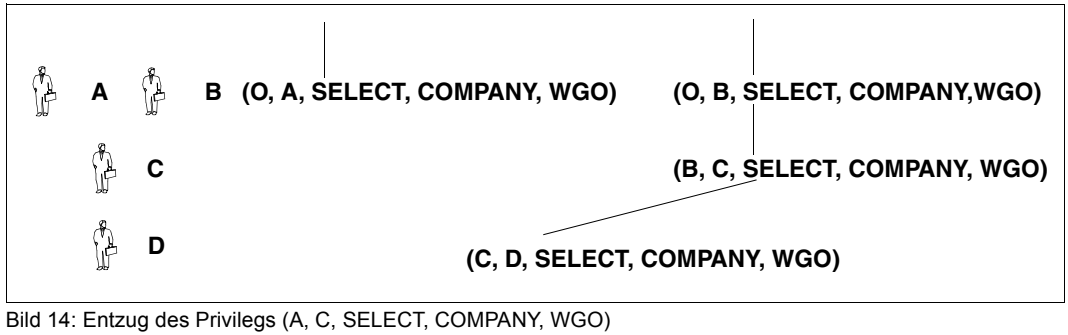


Bild 14: Entzug des Privilegs (A, C, SELECT, COMPANY, WGO)

Es würde kein abgetrenntes Privileg entstehen, da Benutzer C weiterhin das von B erteilte Privileg (B, C, SELECT, COMPANY, WGO) besitzt. Erst beim Versuch von B, C dieses Privileg zu entziehen, würde der entsprechende REVOKE (`REVOKE SELECT ON company FROM C RESTRICT`) abgewiesen. Wäre der REVOKE erfolgreich, hätte dies ein abgetrenntes Privileg (C, D, SELECT, COMPANY, WGO) zur Folge.

Das nächste Beispiel geht von folgender Situation aus (siehe Bild 15): Zwei Benutzer C1 und C2 haben dasselbe Privileg von zwei verschiedenen Benutzern erhalten (C1 von Grantor A, C2 von Grantor B). C1 erteilte daraufhin dem Benutzer D das Privileg (C1, D, SELECT, COMPANY, WGO), C2 erteilte D das Privileg (C2, D, SELECT, COMPANY, WGO).

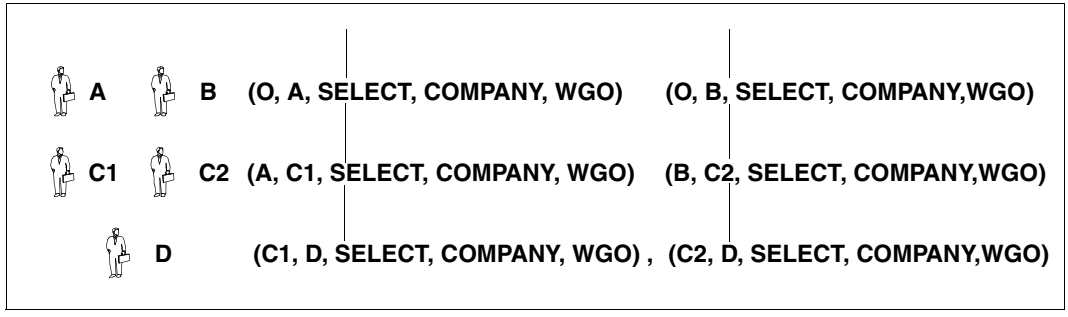


Bild 15: Privilegien (C1, D, SELECT, COMPANY, WGO) und (C2, D, SELECT, COMPANY, WGO)

Der Versuch von A, dem Benutzer C1 das Privileg (A, C1, SELECT, COMPANY, WGO) mit `REVOKE SELECT ON company FROM C1 RESTRICT` zu entziehen, wird abgewiesen, da andernfalls das dem Benutzer D von C1 erteilte Privileg (C1, D, SELECT, COMPANY, WGO) abgetrennt wäre. In Bild 16 ist dargestellt, zu welchem Ergebnis die (unzulässige) Ausführung des REVOKE führen würde.

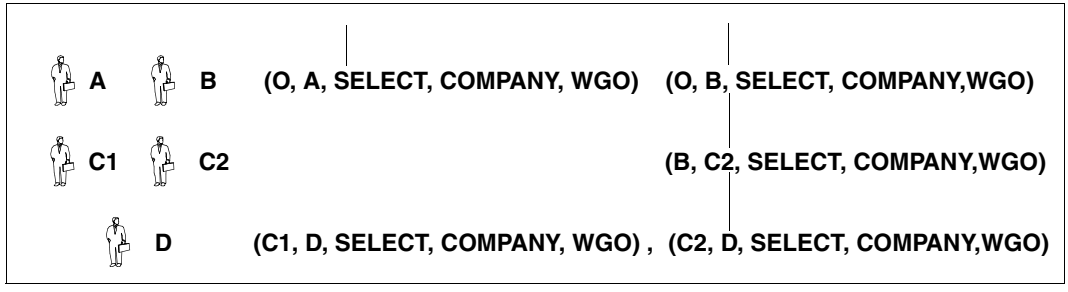


Bild 16: Hypothetische Situation: abgetrenntes Privileg (C1,D,SELECT,COMPANY,WGO)

Die beiden letzten Beispiele (Bild 13 und Bild 14 sowie Bild 15 und Bild 16) verdeutlichen nochmals, warum ein REVOKE ... RESTRICT immer dann abgewiesen wird, wenn seine Ausführung zu einem abgetrennten Privileg führen würde.

Neben dem Begriff „abgetrennte“ Privilegien gibt es die Begriffe „abgetrennte“ Referenzbedingungen und „abgetrennte“ Views.

Vermeidung abgetrennter Referenzbedingungen

Der Begriff „abgetrennte“ Referenzbedingungen soll an folgender Situation erläutert werden.

Der Eigentümer A einer Basistabelle T möchte einem Benutzer B ein zuvor mit GRANT für diese Tabelle erteiltes Tabellen-Privileg REFERENCES mit REVOKE ... RESTRICT entziehen, obwohl eine von diesem Benutzer definierte Referenzbedingung für Spalten der Tabelle noch nicht mit ALTER TABLE ... DROP CONSTRAINT gelöscht worden ist. Die REVOKE ... RESTRICT-Anweisung wird abgewiesen.

Wäre die REVOKE ... RESTRICT-Anweisung erfolgreich, dann gäbe es noch eine Referenzbedingung auf der Tabelle T, für die das zugehörige Tabellen-Privileg REFERENCES nicht mehr existiert. Die Referenzbedingung wäre somit „abgetrennt“.

Mit einem entsprechenden REVOKE ... CASCADE kann Eigentümer A dagegen Benutzer B das REFERENCES-Privileg auch dann entziehen, wenn noch eine Referenzbedingung auf Spalten der Tabelle T definiert ist. Der REVOKE ... CASCADE führt implizit eine ALTER TABLE...DROP CONSTRAINT-Anweisung für diese Referenzbedingung aus.

Vermeidung abgetrennter Views

Neben abgetrennten Privilegien und abgetrennten Referenzbedingungen gibt es auch den Begriff des „abgetrennten“ Views. Der Begriff soll an folgender Ausgangssituation erläutert werden:
Der Eigentümer eines Schemas A vergibt als Grantor das Tabellen-Privileg SELECT für bestimmte Tabellen seines Schemas an den Eigentümer des Schemas B. Der Eigentümer von B erzeugt mit CREATE VIEW einen View V_B, dem diese Tabellen zu Grunde liegen.

Der Eigentümer von A möchte nun mit der REVOKE ... RESTRICT-Anweisung dem Eigentümer von B das Privileg SELECT entziehen. Die REVOKE ... RESTRICT-Anweisung wird jedoch abgewiesen. Wäre der REVOKE ... RESTRICT erfolgreich, dann könnte der Eigentümer von B, d.h. der Eigentümer von V_B, weiterhin über V_B auf die V_B zu Grunde liegenden Tabellen des Schemas A zugreifen, obwohl der Eigentümer von B für diese Tabellen das Tabellen-Privileg SELECT nicht mehr besitzt und so nicht mehr direkt auf die Tabellen zugreifen darf und deshalb nicht einmal mehr den View V_B definieren dürfte.

Es gäbe somit eine View-Definition, obwohl für die entsprechende CREATE VIEW-Anweisung keine Berechtigung mehr besteht. Der Definition des Views V_B ist somit die Berechtigung entzogen, der View ist „abgetrennt“. Allgemein heißt ein View abgetrennt, wenn dem Eigentümer des View das Tabellen-Privileg SELECT für eine der dem View zu Grunde liegenden Tabellen entzogen ist.
Eine REVOKE ... RESTRICT-Anweisung, die zu einem abgetrennten View führen würde, wird stets abgewiesen. Vor jedem REVOKE ... RESTRICT sollte der Benutzer daher sicherstellen, dass alle Views mit der SQL-Anweisung DROP VIEW gelöscht wurden, für deren Definition ein Privileg SELECT Voraussetzung war, das nun mit REVOKE entzogen werden soll.

Entzieht im obigen Beispiel der Eigentümer von A dem Eigentümer von B das Privileg mit einem REVOKE ... CASCADE, dann werden alle abhängigen Views gelöscht. Der REVOKE ... CASCADE entzieht nämlich nicht nur dem Eigentümer von B das SELECT-Privileg für die Tabellen seines Schemas A, sondern führt implizit die folgende Anweisung durch:

```
DROP VIEW B.VB CASCADE
```

Dadurch werden der View V_B und alle Views gelöscht, die den View V_B in ihrer Definition verwenden.

6.4 Zugriffsschutz in Verbindung mit dem Viewkonzept

Das Viewkonzept unterstützt den Zugriffsschutz auf komfortable Weise.

Der Grundgedanke des Viewkonzepts besteht darin, definierten Benutzerkreisen nur die Informationen zur Verfügung zu stellen, die diese benötigen.
In Verbindung mit dem im Abschnitt „Zugriffsschutz in SQL durch Privilegien“ auf Seite 176 beschriebenen Zugriffsschutz durch Privilegien lässt sich gleichzeitig sicherstellen, dass die einzelnen Benutzerkreise jeweils nur Zugang zu solchen Informationen haben, die für diese bestimmt sind. Durch Viewkonzept und Privilegien bietet SESAM/SQL flexible und umfassende Möglichkeiten zur Realisierung des Datenschutzes und trägt so gleichzeitig gezielt dem Informationsbedürfnis der Benutzer wie auch Datenschutzaspekten Rechnung.

Beispielsweise kann es sinnvoll sein, auf einer Tabelle, die Daten aller Mitarbeiter enthält, einen View zu erzeugen, der nur die Spalten Vorname, Nachname und Abteilung enthält. Für diesen View kann dann allen Mitarbeitern das Privileg SELECT eingeräumt werden. Die SELECT-Berechtigung für einen zweiten View, der auch Spalten wie z.B. Gehalt, Geburtsdatum, Adresse usw. enthält, sollte dagegen nur einem ausgewählten Personenkreis, z.B. Mitarbeitern der Personalabteilung, gewährt werden. Noch restriktiver ist zu verfahren, wenn für einzelne Spalten ein Privileg zum Ändern von Spalten vergeben werden soll.
Außerdem ist es denkbar, mit Hilfe von abteilungsspezifischen Views das allgemeine SELECT-Privileg für alle Mitarbeiter dahingehend weiter einzuschränken, dass ein Mitarbeiter nur Daten von Kollegen seiner eigenen Abteilung einsehen kann. In diesem Fall würde man das Privileg abteilungsspezifisch für den jeweiligen View vergeben.

6.5 Kennwortschutz mit SEPA für CALL-DML-Tabellen

CALL-DML-Tabellen kann der Anwender durch Kennwörter gegen unberechtigten Zugriff schützen. Dafür steht das Dienstprogramm SEPA zur Verfügung (siehe Handbuch „Datenbankbetrieb“).

Für eine kennwortgeschützte Tabelle muss der Anwender bei jeder CALL-DML-Anweisung das entsprechende Kennwort angeben. Bei der Analyse der Anweisung prüft SESAM/SQL dann die Berechtigung des Kennworts.

Zuerst wird eine Tabelle oder ein Ausschnitt aus einer Tabelle (logische Datei) eröffnet, indem ein gültiges Kennwort in der Open-Anweisung angegeben wird.
Überschreitet der Anwender bei einer Open-Anweisung im Dialog die maximal zulässige Anzahl von Kennwortverstößen, so wird er vom System gesperrt. Nur der Systemverwalter kann die Sperre wieder aufheben.

Die Tabelle oder die logische Datei kann nach dem Öffnen mit den CALL-DML-Anweisungen bearbeitet werden, wenn dasselbe Kennwort angegeben wird.
Das Kennwort bestimmt dabei, auf welche Daten mit welcher Zugriffsart zugegriffen werden darf. Wird versucht, auf ein Attribut zuzugreifen, für das keine Berechtigung besteht, dann verhält sich SESAM/SQL, als ob das Attribut in der Datenbank nicht existieren würde und weist die Anweisung mit Status zurück.
CALL-DML-Anweisungen mit ungültigem Kennwort werden ebenfalls mit Status abgewiesen.

SEPA bietet folgende Funktionen:

- Ausführen aller Wartungstätigkeiten am Kennwortkatalog:
  Zugriffsrechte ändern, entziehen oder neu vergeben.
- Ausgeben der Informationen über alle Kennwörter (Name, Zugriffsrechte).

Der Kennwortkatalog einer kennwortgeschützten CALL-DML-Tabelle wird nicht gesondert gesichert.

6.6 Zugriffsschutz durch Datenverschlüsselung

Mit den kryptografische Funktionen ENCRYPT und DECRYPT von SESAM/SQL können Sie sensitive Daten einer Datenbank verschlüsseln und entschlüsseln.

Sensitive Daten werden durch die Verschlüsselung vor unbefugtem Zugriff geschützt. Nur die Benutzer, die den sogenannten „Schlüssel“ („key“) kennen, können die Daten entschlüsseln. Auch sensitive Daten einer Datenbank, die in einer nicht-sicheren Umgebung betrieben wird, können so geschützt werden.

SESAM/SQL nutzt den Advanced Encryption Standard (AES) von Rijndael mit einem Schlüssel mit 128 Bit (16 Bytes) im Electronic Codebook Mode (ECM). Allgemeine Informationen zum AES finden Sie im Internet unter der Adresse: *http://csrc.nist.gov/*.

Detaillierte Informationen zu den kryptografischen Funktionen von SESAM/SQL finden Sie im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“.

Schlüssel-Verwaltung

Die Sicherheit verschlüsselter Daten hängt hauptsächlich von der Sicherheit der verwendeten Schlüssel ab. Schlüssel müssen redundant und an unterschiedlichen Stellen, ggf. auch auf unterschiedlichen Medien aufbewahrt werden. Ihr Aufbewahrungskonzept ist Teil des Sicherheitskonzepts.

Der AES verwendet denselben Schlüssel zum Verschlüsseln und zum Entschlüsseln (symmetrisches Verschlüsselungsverfahren). Die Kenntnis des Schlüssels ermöglicht es direkt, sensitive Daten zu entschlüsseln, weil nur der Schlüssel geheim ist, nicht aber das Verschlüsselungsverfahren. Der Schlüssel kann aber auch nicht beliebig oft gewechselt werden, da alle verschlüsselten Daten entschlüsselt und mit dem neuen Schlüssel wieder verschlüsselt werden müßten.

Bei Verlust des Schlüssels können verschlüsselte Daten nicht mehr entschlüsselt werden. SESAM/SQL hinterlässt aus Sicherheitsgründen auch keine interne Spur des Schlüssels oder von unverschlüsselten Werten.

Zum Lesen verschlüsselter Daten aus Sicherungsbeständen müssen ggf. „alte“ Schlüssel aufbewahrt werden.

Speicherung verschlüsselter Daten

Verschlüsselte Daten werden in SESAM/SQL nur in ihrer verschlüsselten Form gespeichert. Dies gilt sowohl für die Spaces der Datenbank, als auch für Indizes, Logging-Dateien (DA-LOG, TA-LOG, WA-LOG) und die temporären Dateien der Utility-Funktionen.

ACHTUNG!
Im Hauptspeicher des DBH und der Client-Anwendungen sind jedoch (nach einer Benutzung von ENCRYPT oder DECRYPT) sowohl die unverschlüsselten Daten als auch der Schlüssel im Klartext sichtbar. Speicherauszüge (Dumps) solcher Bereiche enthalten schutzwürdige Daten und müssen mit besonderer Vorsicht behandelt werden.

Nach Beendigung des DBH und der Client-Anwendungen kann auf die Hauptspeicherbereiche nicht mehr zugegriffen werden. Dumps und Cursor-Dateien (siehe Seite 306, Dateinamen `SES*.CURSOR.*`) sollten nicht nur logisch, sondern auch physikalisch gelöscht werden (z.B. mit dem BS2000-Kommando DELETE-FILE ...,OPTION = *DESTROY-ALL).

Auf der BS2000-Benutzerkennung, in der der DBH läuft, können (z.B. durch Administrationseingriffe) Dateien (z.B. CO-LOG-Datei, Traces, Dumps) entstehen, in denen Schlüssel bzw. die verschlüsselten Daten in entschlüsselter Form wiedergefunden werden können. Daher sollten diese BS2000-Benutzerkennung und auch der Administrationszugang zum DBH geschützt werden.

Auch andere Analysewerkzeuge können Schlüssel und Daten im Klartext sichtbar machen und speichern. Sie verdienen besondere Beachtung. Zur Analyse können Sie einen Mitschnitt der SQL-Anweisungen und ihrer Benutzervariablen mit dem Dienstprogramm SESCOSP erstellen, siehe Handbuch „Datenbankbetrieb“.

NULL-Werte

NULL-Werte werden durch Verschlüsselung wieder zu NULL-Werten.

Privilegien und Verschlüsselung

Privilegien erlauben den Zugang zu Tabellen, zu Spalten und (über Views) zu Zeilen einer Tabelle, siehe Abschnitt „Zugriffsschutz in SQL durch Privilegien“ auf Seite 176. Sie beschreiben auch die zulässigen Operationen (z.B. INSERT) auf diesen Objekten. Privilegien haben einen größeren Funktionsumfang als Verschlüsselung, sie decken jedoch nicht alle Sicherheitsaspekte ab.

Die Verschlüsselung sensitiver Daten ergänzt die Privilegien, z.B. in folgenden Fällen:

- Privilegien können den Zugriff auf sensitive Daten erlauben.
  Bestimmte Rollen (z.B. der universelle Benutzer oder der Eigentümer einer Tabelle) besitzen alle Privilegien für ein Objekt.
  Mit Verschlüsselung können solche Benutzer sensitive Daten jedoch nur dann entschlüsseln, wenn sie auch den Schlüssel kennen.
  Mit ENCRYPT() können einzelne Spalten einer Tabelle oder sogar einzelne Werte gegen unbefugten Zugriff geschützt werden.
- Privilegien kontrollieren den Zugang nur im SQL-System.
  Sie verhindern keine Zugriffe mit anderen Mitteln (z.B. dem BS2000-Kommando SHOW-FILE). Verschlüsselung hingegen verhindert unbefugtes Lesen mit beliebigen anderen Mitteln.
- Privilegien gelten für alle SQL-Anweisungen einer Transaktion.
  Bei Verschlüsselung mit unterschiedlichen Schlüsseln können sensitive Daten mit unterschiedlichen Sicherheitsanforderungen auch unterschiedlich verschlüsselt werden, auch innerhalb ein und derselben Transaktion.
- Bei Privilegien identifiziert sich ein Anwender durch die verwendete Anwendung oder den verwendeten Zugang zum Betriebssystem. SESAM/SQL kann z.B. zwei Anwender mit demselben Zugang nicht unterscheiden. Bei Verschlüsselung können sie sich aber durch die Kenntnis von Schlüsseln unterscheiden.

6.7 Schutz personenbezogener Daten durch Anonymisierung

Die gesetzlichen Bestimmungen zum Datenschutz verlangen, dass Personendaten

- nur zu dem Zweck verwendet und verarbeitet werden, zu dem sie erhoben wurden
- im möglichste kleinem Umfang gehalten werden
- nicht länger aufbewahrt werden, als für die reguläre Verarbeitung benötigt

Bei Abweichungen von diesen Anforderungen müssen Personendaten anonymisiert werden, so dass eine Abbildung auf die natürliche Person, zu der die Daten ursprünglich gehört haben, nicht mehr möglich ist.

Zitat: §3 (6) Datenschutzgesetz der Bundesrepublik Deutschland:
„**Anonymisieren** ist das Verändern personenbezogener Daten derart, dass die Einzelangaben über persönliche oder sachliche Verhältnisse nicht mehr oder nur mit einem unverhältnismäßig großen Aufwand an Zeit, Kosten und Arbeitskraft einer bestimmten oder bestimmbaren natürlichen Person zugeordnet werden können."

Eine verbreitete Praxis ist es, aus Produktivdatenbanken Testdatenbanken zu erzeugen. Aus Sicht des Datenschutzes müssen bei diesem Vorgehen aber personenbezogene Daten geschützt werden.

Mit der Utility Funktion ALTER DATA FOR TABLE stellt SESAM/SQL eine Funktion zur Verfügung, die die Anonymisierung von Daten unterstützt, so dass keine Rückschlüsse auf den ursprünglichen Zusammenhang möglich sind. Diese anonymisierten Testdaten dürfen dann auch aus Datenschutzsicht weiter verwendet werden.

Die Anonymisierung der Daten wird nicht protokolliert. Es gibt keine Funktion, um die Anonymisierung der Daten rückgängig zu machen. Aus einem Vergleich der Daten vor und nach der Anonymisierung kann der angewendete Algorithmus nicht ermittelt werden.

Die Daten selbst werden dabei nicht verändert, nur die Zuordnung der Spaltenwerte zu den einzelnen Zeilen der Tabelle wird verändert. Das Wertespektrum und auch die Häufigkeitsverteilung der einzelnen Spaltenwerte bleibt erhalten.

Die Zuordnung der Spaltenwerte zu den einzelnen Zeilen der Tabellen wird für jede Spalte und bei jedem Aufruf der Funktion unterschiedlich durchgeführt. Logische zusammenhängende Spalten können auch gemeinsam vertauscht werden.

Beispiel

Die Spaltenwerte der Tabelle *personal* werden vertauscht.
Dabei bleibt der logische Zusammenhang der Spalten Anrede, Vorname, Geschlecht sowie der Spalten Ort und Postleitzahl erhalten.

```
ALTER DATA FOR TABLE personal
SHUFFLE VALUES FOR COLUMN (Anrede, Vorname, Geschlecht),Nachname,
                          (Postleitzahl, Ort), Strasse, Hausnummer,
                          Telefon, Geburtsdatum, Geburtsort
```

6.8 Protokollierung sicherheitsrelevanter Ereignisse mit SAT

SESAM/SQL protokolliert sicherheitsrelevante Ereignisse mit Hilfe der Komponente SAT (Security Audit Trail) des Softwareprodukts SECOS (Security Control System).

Dazu werden, wenn die SAT-Protokollierung in SESAM/SQL eingeschaltet ist, von SESAM/SQL sogenannte Protokolldatensätze (SATLOG-Sätze) an SAT übergeben. SAT trägt diese SATLOG-Sätze in eine geschützte Protokolldatei (SATLOG-Datei) ein. Die SATLOG-Datei kann mit Hilfe des SAT-Auswerters SATUT analysiert werden. SATUT erzeugt sinnvoll aufbereitete SAT-Protokolldateien und/oder Ergebnislisten. Informationen zu SAT und dem SAT-Auswerteprogramm SATUT finden Sie im SECOS-Handbuch „Security Control System - Beweissicherung“.

SESAM/SQL stellt dem SESAM-Systemverwalter folgende Möglichkeiten zur Verfügung, mit denen er für SESAM/SQL die SAT-Protokollierung ein- und ausschalten kann:

- DBH-Option SECURITY, siehe Handbuch „Datenbankbetrieb“, Kapitel „DBH-Startanweisungen und -Optionen“
- Administrationsanweisung SET-SAT-SUPPORT bzw. OPT,SAT, siehe Handbuch „Datenbankbetrieb“, Kapitel „DBH- und SESDCN-Administration“

Um eine Protokollierung der SESAM-Ereignisse zu erreichen, müssen sowohl die SAT-Protokollierung in SESAM/SQL als auch die SESAM-Ereignisse bei der SAT-Preselection in SAT eingeschaltet sein.

Für folgende Ereignisse werden (bei eingeschalteter SAT-Protokollierung) SATLOG-Sätze von SESAM/SQL an SAT übergeben:

- Start oder Ende einer SESAM-DBH- oder Service-Task
- Ende eines Vorganges
- Beeinflussung der DBH-Session durch Administration
- Manipulation der Datenbankstruktur mittels DDL oder SSL
- Durchführung von Utility-Anweisungen
- Änderungen in Benutzerzugängen und Zugriffsberechtigungen

DML-Zugriffe werden **nicht** mit SAT protokolliert.

Die SATLOG-Sätze von SESAM/SQL sind im Anhang des Handbuchs „Datenbankbetrieb“, Abschnitt „Aufbau der Protokolldatensätze für SAT“ beschrieben.

7 Sicherungskonzept

Das Sicherungskonzept von SESAM/SQL besteht aus folgenden Komponenten:

- dem Transaktionskonzept, das die Datenkonsistenz im laufenden Betrieb garantiert
- dem Recovery-Konzept, bei dem im Fehlerfall mit verschiedenen Maßnahmen die Konsistenz der Daten wiederhergestellt wird

Dieses Kapitel beschreibt

- das Transaktionskonzept
- die Transaktionssicherung als Grundlage für Transaktionsrecovery und Wiederanlauf
- die Behandlung von Transaktionen beim Wiederanlauf
- mögliche Fehlersituationen und geeignete Recovery-Maßnahmen
- das Media-Recovery, das auf SESAM-Sicherungsbeständen und Logging aufbaut
- die Benutzung von Replikaten
- die sicherungsspezifischen Dateien

7.1 Transaktionskonzept

Eine Transaktion ist eine Folge von zusammengehörigen Anweisungen, die eine Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt.

Eine Transaktion wird entweder vollständig oder überhaupt nicht ausgeführt.

Beispiel

Es sollen 1000 Euro von Konto 1789 auf Konto 1564 überwiesen werden.

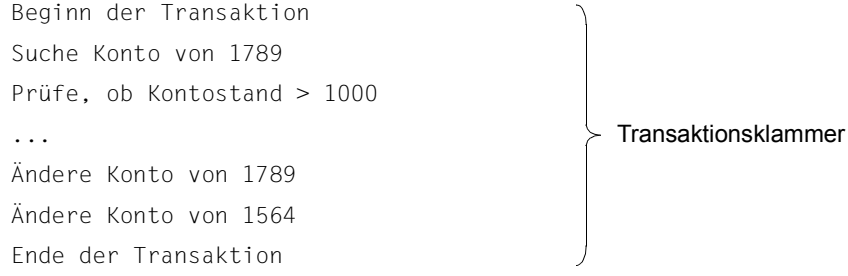
Folgende Zugriffe sind notwendig:

– Lesen des Kontostands von Konto 1789

– Prüfen, ob Kontostand 1789 größer 1000 Euro ist

– Ändern des Kontostandes bei Konto 1789 und 1564.

Die Transaktion hat folgenden Aufbau:



7.1.1 Transaktion im Anwenderprogramm

Transaktionen werden unterschiedlich eröffnet und geschlossen:

SQL-Transaktion:
: Nach dem Programmstart oder dem Beenden der vorhergegangenen Transaktion beginnt die Transaktion mit der ersten transaktionseinleitenden SQL-Anweisung. Die Anweisung zum Festschreiben einer Transaktion ist die COMMIT WORK-, zum Zurücksetzen einer Transaktion die ROLLBACK WORK-Anweisung.

CALL-DML-Transaktion:
: Beginn und Ende einer Transaktion definieren Sie durch CALL-DML-Anweisungen in Ihrem Programm. Die Transaktion beginnt mit der Anweisung „Beginn Transaktion“ (BTA) und endet mit der Anweisung „Ende Transaktion“ (ETA) bzw. durch internes Zurücksetzen der Transaktion durch den DBH.

In einer SQL-Transaktion dürfen SQL- und CALL-DML-Anweisungen nicht gemischt vorkommen.

In einer CALL-DML-Transaktion dürfen aber bestimmte SQL-Anweisungen eingegeben werden. Ausnahmen sind die Anweisungen COMMIT WORK und ROLLBACK WORK sowie alle SQL-Anweisungen zum Abfragen und Ändern, die auch in einer SQL-Transaktion nicht verwendet werden dürfen (siehe Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“).

Anwendungen mit linked-in DBH können nur reine CALL-DML- oder SQL-Transaktionen enthalten.

Eine Transaktion kann Anweisungen an eine oder mehrere Tabellen enthalten.

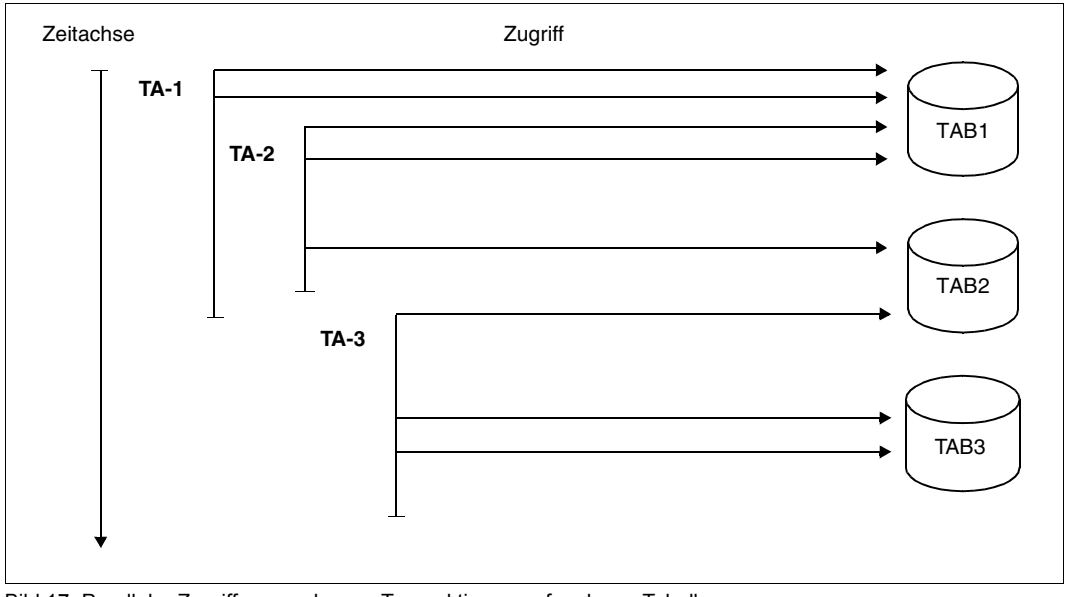


Bild 17: Paralleler Zugriff von mehreren Transaktionen auf mehrere Tabellen

SESAM/SQL ermöglicht den parallelen Zugriff von maximal 32767 Transaktionen auf je 1 bis 254 Datenbanken. Bei CALL-DML sind maximal 254 Tabellen möglich (siehe OLD-TABLE-CATALOG, Handbuch „Datenbankbetrieb“).

7.1.2 Sperrkonzept in SESAM/SQL-Transaktionen

Das Sperrkonzept gewährleistet die Konsistenz der bearbeiteten Daten.

CALL-DML: In einer Transaktion werden alle Sätze exklusiv gelesen, falls die zugehörige logische Datei mit Open-Modus „X“ eröffnet wurde und der Zusatz „readnolock“ nicht gegeben wurde.
Ein shared Lesen erfolgt nur, falls die zugehörige logische Datei mit Open-Modus „R“ eröffnet wurde.

SQL: Leseoperationen erfolgen im Shared-Modus, Änderungsoperationen im Exklusiv-Modus.

Auf diese Weise sind Datenverfälschungen ausgeschlossen.

Modifikationen bei CALL-DML

Bei der Datenwiedergewinnung der CALL-DML erlaubt SESAM/SQL folgende Modifikationen des Sperrkonzepts (siehe Handbuch „CALL-DML Anwendungen“):

- Lesen ohne zu sperren (kann zu „non-repeatable read“ führen)
- Ignorieren der exklusiven Sperre (kann zu „dirty read“ führen).

Modifikationen bei SQL

Das Sperrkonzept sowie seine Modifikationen sind bei SQL auf Isolationslevel abgebildet. Der SQL-Anwender legt für jede Transaktion explizit oder implizit einen Isolationslevel fest:

- implizit zum Übersetzungszeitpunkt bzw. durch die Konfigurationsdatei
- explizit durch die SET TRANSACTION-Anweisung oder das Pragma ISOLATION LEVEL.

Dadurch wird gleichzeitig der Grad der Transaktionsparallelität bestimmt (siehe Abschnitt „SQL-Transaktion“ auf Seite 134).

Wartezustände bei Transaktionen

Versucht eine Transaktion auf einen Satz zuzugreifen, der von einer anderen Transaktion gesperrt ist, so kann sie die Anweisung zunächst nicht ausführen: Die Transaktion wird in einen Wartezustand versetzt. Gleichzeitig wird in der Tabelle der offenen Transaktionen vermerkt, welche Transaktion die Sperre des Satzes ausgelöst hat.

Tabelle der offenen Transaktionen	
Transaktion	gesperrt durch Transaktion
A	
B	A
C	B
D	A

Tabelle 45: Transaktionen im Wartezustand

Die wartende Transaktion kann erst dann mit der Bearbeitung fortfahren, wenn die sperrende Transaktion abgeschlossen oder zurückgesetzt ist. Zu diesem Zweck überwacht der SESAM/SQL-DBH die wartenden Transaktionen und reaktiviert sie, wenn die sperrenden Transaktionen beendet sind.

Mit der untergeordneten DBH-Option TRANSACTION-SECURITY kann das Verhalten des DBH gegenüber sperrenden Transaktionen und das Eskalieren von Transaktionssperren beeinflusst werden. Während des laufenden Betriebs können diese Parameter mit der Administrationsanweisung MODIFY-TRANSACTION-SECURITY geändert werden (siehe Handbuch „Datenbankbetrieb“).

Es können sich zwei Situationen ergeben, in denen eine automatische Reaktivierung nicht möglich ist bzw. unnötig lange verzögert wird: Deadlock- und Longlock-Situationen.

Deadlock-Situationen

Ein Deadlock liegt immer dann vor, wenn sich zwei oder mehrere Transaktionen gegenseitig sperren. In diesem Fall kann keine der Transaktionen abgeschlossen werden.

Mit der Administrationsanweisung SET-SESSION-DIAGNOSIS können Sie sich bei Auftreten eines Deadlock zusätzliche Informationen über die Benutzer und die betroffenen Objekte auf SYSLST ausgeben lassen, siehe Handbuch „Datenbankbetrieb“.

Die folgende Tabelle zeigt den Deadlock von 3 Transaktionen. Dabei besteht zwischen der Art des Zugriffs und der Anzahl der E/A-Vorgänge folgender Zusammenhang:
Ein Lesezugriff auf einen Datenblock entspricht einem E/A-Vorgang, ein Schreibzugriff entspricht zwei E/A-Vorgängen.

Tabelle der offenen Transaktionen		
Transaktion	gesperrt durch Transaktion	E/A-Vorgänge
A	C	100
B	A	20
C	B	60
D		5

Tabelle 46: Deadlock von 3 Transaktionen

Der SESAM/SQL-DBH löst Deadlock-Situationen automatisch auf.

Jedesmal, wenn eine Transaktion gesperrt wird, prüft der SESAM/SQL-DBH, ob diese Sperre einen Deadlock auslöst.

Ist dies der Fall, setzt der SESAM/SQL-DBH die Transaktion zurück, die am Deadlock beteiligt ist und bis zu diesem Zeitpunkt die wenigsten E/A-Vorgänge hatte (in Tabelle 46 ist dies Transaktion B). Die entsperrten Transaktionen können darauf die Verarbeitung fortsetzen. In Tabelle 47 ist dies Transaktion C.

Nach Auflösen der Deadlock-Situation ergibt sich folgender aktueller Stand:

Tabelle der offenen Transaktionen		
Transaktion	gesperrt durch	E/A-Vorgänge
A	C	100
C		60
D		5

Tabelle 47: Transaktionstabelle nach Auflösen der Deadlock-Situation

Aufgrund des Rücksetzkriteriums „wenigste E/A-Vorgänge“ wird stets die Transaktion mit dem geringsten Rücksetzaufwand zurückgesetzt.

Longlock-Situationen

Der SESAM/SQL-DBH ermittelt Transaktionen, die für lange Zeit die Verarbeitung unterbrechen und Betriebsmittel für andere Transaktionen sperren. Solche Transaktionen setzt der SESAM/SQL-DBH automatisch zurück. Durch ein Rücksetzkriterium legt der Systemverwalter fest, wie lange inaktive Transaktionen andere Transaktionen sperren dürfen (LOCK-TIME-Parameter der DBH-Option TRANSACTION-SECURITY bzw. der Administrationsanweisung MODIFY-TRANSACTION-SECURITY, siehe Handbuch „Datenbankbetrieb“).

7.2 Transaktionssicherung

Die Transaktionssicherung umfasst Mechanismen zur automatischen Fehlerbehandlung und stellt im Fehlerfall die Konsistenz der Datenbestände über Rücksetzmechanismen wieder her. Dieser Vorgang wird als Transaktionsrecovery bezeichnet.
Außerdem bildet die Transaktionssicherung die Grundlage für die Recovery-Maßnahmen externer und interner Wiederanlauf.
Die Transaktionssicherung gewährleistet für die Datenbank physikalische und logische Konsistenz.

Physikalische Konsistenz

Die Transaktionssicherung gewährleistet die physikalische Konsistenz, indem der SESAM/SQL-DBH den physikalischen Datenbankblock vor der ersten Änderung als physikalisches Before-Image (PBI) im Puffer zwischenspeichert und bei vorzeitiger Verdrängung des Datenblockes in die Wiederanlaufsicherungsdatei sichert (WA-LOG-Datei). Wenn der Datenblock vor dem Transaktionsende nicht vorzeitig verdrängt wird, dann wird kein PBI geschrieben.
Das Abspeichern der After-Images (AI), die durch Änderungen entstanden sind, dient ebenfalls der Sicherung der physikalischen Konsistenz.

Logische Konsistenz

Die logische Konsistenz wird sichergestellt durch Schreiben der zu ändernden Teile eines Satzes als logisches Before-Image (LBI) in die Transaktionssicherungsdatei.

Konsistenzpunkte

Bei jeder Anweisung für das Ende der Transaktion setzt der DBH einen Konsistenzpunkt in der jeweils aktuellen Transaktionssicherungsdatei. Beenden sich mehrere parallele Transaktionen kurz hintereinander, so wird für diese Transaktionen ein gemeinsamer Sammelkonsistenzpunkt (Group Commit) geschrieben.

Beständigkeit der Wirkung einer Transaktion

Bei Transaktionsende werden die physikalischen und logischen Before-Image-Blöcke der abschließenden Transaktion gleichzeitig freigegeben. Die Transaktion ist nun nicht mehr rücksetzbar und demzufolge beständig. Die von der Transaktion betroffenen Datenbanken sind in Bezug auf diese Transaktion zusammen mit den After-Images in einem logisch konsistenten Zustand.

Transaktionssicherungsdateien

Die Transaktionssicherung (TA-Sicherung) ist die Voraussetzung für ein sessionbezogenes Wiederanlaufverfahren.

Für den Betrieb mit Transaktionssicherung legt der SESAM/SQL-DBH die verschiedenen Transaktionssicherungsdateien an:

TA-LOG-Dateien

Es werden jeweils zwei TA-LOG-Dateien im Wechsel beschrieben. In diesen zwei Dateien sind Informationen zur Absicherung der DBH-Session gespeichert. Sie sind jeweils einem SESAM/SQL-DBH eindeutig zugeordnet. Die TA-LOG-Dateien enthalten u.a.:

- die logischen Before-Images
- die physikalischen After-Images
- Informationen zu den Konsistenzpunkten für den automatischen Wiederanlauf nach Systemausfall

WA-LOG-Datei

In dieser Datei sind Informationen zur Steuerung und Absicherung der DBH-Session und des DBH-Wiederanlaufs gespeichert (siehe Abschnitt „DBH-spezifische Dateien“ auf Seite 302). Jedem SESAM/SQL-DBH ist eine WA-LOG-Datei eindeutig zugeordnet.
Die WA-LOG-Datei enthält u.a.:

- die DBH-Optionen, die der DBH für den Wiederanlauf übernimmt
- Liste der am Restart beteiligten Spaces
- die physikalischen Before-Images
- Informationen, die den Fortschritt des Wiederanlaufs beschreiben
- Information zur Synchronisation mit openUTM

DDL-TA-LOG-Datei

Diese Datei ist bei SQL-Anwendungen dem betroffenen Space zugeordnet. Sie dient zur Absicherung von langlaufenden DDL- und SSL-Anweisungen, die in der Servicetask ablaufen, sowie zur Entlastung der TA-LOG-Dateien.

Nach ordnungsgemäßem Ende der Anweisung wird sie wieder gelöscht, wenn sie nicht durch den Anwender angelegt worden war.

SESAM/SQL legt die DDL-TA-LOG-Datei stets auf der DBH-Kennung an, auch wenn die Datenbank auf einer DB-Kennung liegt. Das Anlegen erfolgt auf dem Speichermedium (Platte) und mit der Primär- und Sekundärzuweisung gemäß den Angaben der DDL-TA-LOG-Mediensätze in der Medientabelle.

Für Primär- und Sekundärzuweisung der DDL-TA-LOG-Datei gelten die Standardwerte 1536 (Primärzuweisung) und 384 (Sekundärzuweisung).

7.3 Wiederanlauf

SESAM/SQL führt einen Wiederanlauf durch, wenn der Betrieb nach Fehlerfällen wieder aufgenommen wird.

Bei einem Wiederanlauf setzt der DBH die Transaktionen zurück, die zum Zeitpunkt eines Fehlers offen waren. Alle beteiligten Datenbanken liegen dann in einem konsistenten Zustand vor. Voraussetzung für den Wiederanlauf ist, dass der DBH mit Transaktionssicherung läuft und die Transaktionssicherungsdateien TA-LOG1, TA-LOG2 und WA-LOG vom Zeitpunkt des Crashes zur Verfügung gestellt werden. Steht eine der Dateien nicht zur Verfügung oder stammen die Dateien von unterschiedlichen Zeitpunkten, ist ein Wiederanlauf nicht möglich. Je nachdem, ob die laufende DBH-Session unterbrochen wird oder nicht, führt der DBH einen externen oder internen Wiederanlauf durch.

- Externer Wiederanlauf:
  Wiederanlauf nach Systemausfall, den der DBH durchführt, wenn er nach einer Abbruch-Session geladen wird. Bei Systemausfall ermöglicht die Transaktionssicherung den automatischen Wiederanlauf des SESAM/SQL-DBH. Die Datenbanken werden behandelt, wie folgt:
  - Mit Hilfe der physikalischen Before Images wird der Stand des letzten Konsistenzpunktes wiederhergestellt. Zu diesem Zeitpunkt noch nicht ausgeführte Schreibaufträge auf die Datenbank werden mit den After Images nachgeholt (physikalische Reparatur).
  - Anschließend werden alle zum Zeitpunkt des letzten Konsistenzpunkts offenen Transaktionen anhand der logischen Before-Images zurückgesetzt (logische Reparatur).
  - Nach dem Wiederanlauf bleiben die Datenbanken geöffnet und können wieder durch Anwenderprogramme bearbeitet werden.
- Interner Wiederanlauf:
  Wiederanlauf nach internen Fehlern geringeren Gewichts. Der DBH führt den internen Wiederanlauf durch, ohne den laufenden Betrieb zu unterbrechen.
  Nach einer Reinitialisierung interner Speicherbereiche werden wie beim externen Wiederanlauf physikalische und logische Reparaturen durchgeführt. Anschließend wird die Session fortgesetzt.

Die Zeitdauer des Wiederanlaufs hängt im Wesentlichen von der Dauer der physikalischen und logischen Reparatur ab. Um die Zeit bis zur Verfügbarkeit des DBH nach dem Wiederanlauf zu verkürzen, haben Sie folgende Steuerungsmöglichkeiten:

- Sie können die Häufigkeit beeinflussen, mit der die After-Image-Blöcke während des normalen Betriebs auf die Datenbank geschrieben werden.
- Sie können das logische Zurücksetzen bis zum Start des Normalbetriebs nach dem Wiederanlauf verzögern.

Je öfter geänderte Blöcke bereits während des Normalbetriebs auf die Datenbank geschrieben werden, desto weniger muss dieses bei einem möglichen Wiederanlauf bei der physikalischen Reparatur mit Hilfe der After Images geschehen. Die Dauer bis zum Normalbetrieb nach einem Abbruch verkürzt sich. Allerdings kann sich zu häufiges Schreiben auf die Datenbank während des Normalbetriebs negativ auf die Performance auswirken. Sie sollten daher die I/O-Raten mit dem Performance-Monitor beobachten und die Einstellungen gegebenenfalls während des laufenden Betriebs anpassen (siehe Handbuch „Datenbankbetrieb“).

Wenn Sie das logische Zurücksetzen bis zum Start des Normalbetriebs verzögern, verlaufen die Rücksetzvorgänge parallel zu den normalen Transaktionen bzw. Aufträgen der Anwender und werden wie im Normalbetrieb synchronisiert. Eine detaillierte Beschreibung dieser Option finden Sie im Handbuch „Datenbankbetrieb“.

7.4 Mögliche Fehlersituationen und geeignete Recovery-Maßnahmen

Beim Arbeiten mit einer SESAM/SQL-Datenbank können folgende Fehlersituationen auftreten:

- Rechnerausfall (z.B. auf Grund unterbrochener Stromversorgung)
- Geräteausfall (z.B. defekte Platte wegen defektem Schreib/Lesekopf)
- Softwarefehler im Betriebssystem ohne Verfälschung der Daten
- Softwarefehler im Betriebssystem mit Verfälschung der Daten
- Softwarefehler im Datenbanksystem ohne Verfälschung der Daten
- Softwarefehler im Datenbanksystem mit Verfälschung der Daten
- Fehler in der Anwendersoftware, die dazu führen, dass falsche oder unvollständige Daten in der Datenbank gespeichert werden

Durch Rechnerausfall verursachte Fehlersituationen sowie Fehlersituationen auf Grund von Softwarefehlern im Betriebssystem oder Datenbanksystem beherrscht SESAM/SQL mit der Transaktionssicherung und der Fähigkeit des SESAM/SQL-DBH zum Wiederanlauf, sofern keine Daten verfälscht oder Dateien des Datenbanksystems zerstört wurden (siehe Abschnitt „Transaktionssicherung“ auf Seite 205 sowie Abschnitt „Wiederanlauf“ auf Seite 207).

Fehlersituationen auf Grund einer defekten Platte beschränkt SESAM/SQL so weit wie möglich auf die Spaces, die auf dieser Platte liegen. Darüber hinaus ermöglicht die Nutzung der BS2000-Funktion DRV (Dual Recording by Volume), Fehlersituationen auf Grund eines Plattenfehlers zu beherrschen und sie vor SESAM/SQL zu verbergen.
Bei Zuschalten einer neuen Platte als Spiegelplatte muss der Datenabgleich, d.h. die Egalisierung mit BS2000-Mitteln erfolgen. Dagegen muss nach einem Systemausfall die Egalisierung von Datenbankbereichen mit unterschiedlichem Stand nicht vom Datenbankverwalter durchgeführt werden, da der Wiederanlauf von SESAM/SQL diese Aufgaben automatisch erledigt.

Grundsätzlich lassen sich alle Ausfall- und Fehlersituationen mit Hilfe des Media-Recovery beherrschen (siehe Abschnitt „Media-Recovery“ auf Seite 211). Wiederanlauffähigkeit des SESAM/SQL-DBH und Nutzung der BS2000-Funktion DRV stellen jedoch die Verfügbarkeit der Anwendung in den jeweils genannten Fehlersituationen schneller wieder her.

In den folgenden Fehlersituationen stellt das Media-Recovery die einzige Möglichkeit dar, eine defekte SESAM/SQL-Datenbank wiederherzustellen:

- Softwarefehler im Betriebssystem mit Verfälschung der Daten
- Softwarefehler im Datenbanksystem mit Verfälschung der Daten
- Fehler in der Anwendersoftware, die zu falschen oder unvollständigen Daten in der Datenbank führen
- Zerstörung von Dateien des Datenbanksystems auf Grund von Hardwarefehlern

7.5 Media-Recovery

Mit Hilfe des Media-Recovery kann der Datenbankverwalter eine defekte SESAM/SQL-Datenbank auch dann reparieren, wenn z.B. nach einem Hardware-Fehler andere Recovery-Maßnahmen nicht zum Erfolg führen.

Das Media-Recovery stützt sich auf folgendes Konzept:

- Zu sorgfältig ausgewählten Zeitpunkten erstellt der Datenbankverwalter SESAM-Sicherungsbestände. Er kann die Sicherungsbestände auf Platte oder mit ARCHIVE bzw. HSMS auf Magnetbandkassette anlegen (Utility-Anweisung COPY, siehe Seite 368).

  Neben der Sicherungseinheit Datenbank unterstützt SESAM/SQL auch kleinere Sicherungseinheiten. Die kleinste Sicherungseinheit ist ein einzelner Anwender-Space. Der Zeitpunkt der Sicherung sowie geeignete Sicherungseinheiten haben wesentlichen Einfluss auf die Dauer der Recovery-Maßnahmen.

- Alle Änderungen der Datenbank, die nach dem Erstellen eines SESAM-Sicherungsbestandes anfallen, protokolliert SESAM/SQL auf Logging-Dateien (CAT-Logging, DA-Logging).

- Die Reparatur der Datenbank bzw. eines Anwender-Space erfolgt durch Zurücksetzen auf einen geeigneten SESAM-Sicherungsbestand und Nachfahren der in den Logging-Dateien (CAT-LOG-Dateien, DA-LOG-Dateien) protokollierten Änderungen auf diesen SESAM-Sicherungsbestand.

7.5.1 BS2000-Benutzerkennungen beim Media-Recovery

Grundsätzlich können SESAM-Sicherungsbestände und Logging-Dateien sowohl in der DBH-Kennung als auch in einer DB-Kennung liegen.

SESAM/SQL versucht immer zuerst, die SESAM-Sicherungsbestände oder Logging-Dateien in der DB-Kennung zu erstellen. Das geht jedoch nur, wenn auf der DB-Kennung die DBH-Kennung als Miteigentümer an den DB-Dateien definiert ist oder der Datenbankverwalter die entsprechenden Dateien mit CREATE-FILE schon in der BS2000-Benutzerkennung angelegt hat, siehe Abschnitt „Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen“ auf Seite 308.

Alle für eine Sicherung benötigten Dateien müssen in einer Kennung, der DB-Kennung oder der DBH-Kennung, angelegt werden können.

Ebenso wird beim RECOVER und REFRESH davon ausgegangen, dass sich die Dateien in einer DB-Kennung befinden. Erst wenn SESAM/SQL dort keine Kopien findet, sucht es in der DBH-Kennung. Analog wird bei den Logging-Dateien verfahren.

Wenn Sicherungsdateien sowohl in der DB- als auch in der DBH-Kennung vorhanden sind, werden die Dateien der DB-Kennung verwendet.

7.5.2 Dateien und Tabellen für das Media-Recovery

Für das Media-Recovery verwendet SESAM/SQL die folgenden datenbank-spezifischen Dateien und im Catalog-Space abgelegten Tabellen (Catalog-Tabellen):

- PBI-Datei (bei Plattenkopie und Online-Sicherung mit ARCHIVE)
  bzw. eine HSMS-Arbeitsdatei (bei Online-Sicherung mit HSMS)
- CAT-REC-Datei
- CAT-LOG-Dateien
- DA-LOG-Dateien
- Catalog-Tabelle RECOVERY_UNITS
- Catalog-Tabelle DA_LOGS

Dateieigenschaften und Speichermedien für nicht vorhandene datenbank-spezifische Dateien legt der Datenbankverwalter in der Medientabelle fest.

7.5.2.1 Medientabelle

Die Medientabelle liegt auf dem Catalog-Space und enthält für jede datenbank-spezifische Dateiart Einträge mit einer Beschreibung der Dateieigenschaften (Größe von Primär- und Sekundärzuweisung, Mehrfachbenutzbarkeit) und Informationen über die Speichermedien, auf denen die Dateien der betreffenden Dateiart angelegt werden sollen. Jeder solche Eintrag heißt Mediensatz.

Medientabelle bearbeiten

Der erste Mediensatz für die CAT-REC-Datei sowie der erste Mediensatz für CAT-LOG-Dateien wird bereits beim Anlegen des Catalog-Space mit der Utility-Anweisung CREATE CATALOG in die Medientabelle aufgenommen (siehe Abschnitt „Catalog-Space der Datenbank anlegen“ auf Seite 341).
Den jeweils ersten Mediensatz für DA-LOG-Dateien und die PBI-Datei bzw. die HSMS-Arbeitsdatei trägt der Datenbankverwalter mit der Utility-Anweisung CREATE MEDIA DESCRIPTION in die Medientabelle ein (siehe Seite 349).

Mit der Utility-Anweisung ALTER MEDIA DESCRIPTION kann der Datenbankverwalter die Medientabelle modifizieren, indem er einzelne Mediensätze neu aufnimmt oder löscht. Außerdem kann er mit dieser Anweisung die Beschreibung der Dateieigenschaften ändern. Mit der Utility-Anweisung DROP MEDIA DESCRIPTION können alle Mediensätze für eine bestimmte Dateiart gelöscht werden (siehe Seite 349).

Auswertung der Medientabelle durch SESAM/SQL

Beim Anlegen einer Datei sucht SESAM/SQL zunächst nach dieser Datei. Bei Logging-Dateien verfährt es nach den entsprechenden Regeln (siehe Abschnitt „BS2000-Benutzerkennungen beim Media-Recovery“ auf Seite 211). Wird die Datei nicht gefunden, wertet SESAM/SQL bei einer datenbank-spezifischen Datei die Medientabelle aus und legt die Datei gemäß den Angaben des ersten Mediensatzes für die betreffende Dateiart an. Gibt es zu dieser Dateiart keinen Eintrag in der Medientabelle, dann versucht SESAM/SQL die Datei immer zuerst in der Katalogkennung einzurichten, die der DB-Kennung zugeordnet ist, siehe Abschnitt „Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen“ auf Seite 308.
Anderenfalls legt SESAM/SQL die Datei in der DBH-Kennung an. Welche Standardwerte SESAM/SQL dabei im Einzelnen verwendet, ist der nachfolgenden Beschreibung der datenbank-spezifischen Dateiarten zu entnehmen.

SESAM/SQL legt solange Dateien auf dem im ersten Mediensatz für die betreffende Dateiart angegebenen Speichermedium an, bis auf dem Gerät kein Platz mehr vorhanden ist. Erst dann wertet SESAM/SQL den zweiten Mediensatz für diese Dateiart aus und legt die nächste Datei dieser Dateiart auf dem im zweiten Mediensatz beschriebenen Speichermedium an. Dieses Verfahren setzt SESAM/SQL solange fort, bis die Medientabelle abgearbeitet, also der letzte Mediensatz für die betreffende Dateiart erreicht ist. Über eine Klausel der Utility-Anweisung CREATE MEDIA DESCRIPTION kann der Datenbankverwalter festlegen, ob SESAM/SQL bei abgearbeiteter Medientabelle am Bedienplatz weitere Angaben zu Speichermedien für die betreffende Dateiart anfordert oder nicht.

Für die PBI-Datei bzw. die HSMS-Arbeitsdatei wird die Medientabelle nur bei COPY ausgewertet. Bei RECOVER einer Online-Sicherung, die mit Hilfe von ARCHIVE erstellt wurde, wird die PBI-Datei auf dem Default-Pubset der DBH-Kennung angelegt.
Wenn ein DVS-Fehler während der Online-Sicherung auftritt, z.B. weil die Datei nicht erweitert werden kann, wird die Sicherung abgebrochen.

7.5.2.2 PBI-Datei

Eine PBI-Datei benötigt SESAM/SQL, um bei der Online-Erstellung von SESAM-Sicherungsbeständen (bei Plattenkopie und Online-Sicherung mit ARCHIVE) mit der Utility-Anweisung COPY einen konsistenten Stand der SESAM-Sicherungsbestände gewährleisten zu können (siehe Seite 368).
Bei Online-Sicherung mit HSMS wird eine HSMS-Arbeitsdatei verwendet.

Inhalt der PBI-Datei bzw. der HSMS-Arbeitsdatei

In der PBI-Datei protokolliert SESAM/SQL die physikalischen Before Images (PBI) von Blöcken, die während des Kopiervorgangs durch parallele SQL-Anweisungen und/oder CALL-DML-Anweisungen geändert werden.

Am Ende der Online-Erstellung eines SESAM-Sicherungsbestandes auf Platte spielt SESAM/SQL die protokollierten PBIs sofort in den SESAM-Sicherungsbestand ein.

Bei der Sicherung auf Magnetbandkassette mit HSMS übernimmt HSMS die Verwaltung der Before Images. Die HSMS-Arbeitsdatei enthält die Before Images von Blöcken, die im zu sichernden Space geändert werden. HSMS führt eine Verwaltung über die während der Dauer der Sicherung geänderten Blöcke und liest den zu sichernden Block von der HSMS-Arbeitsdatei oder vom Space. Nach Abschluß der Sicherung löscht HSMS die Arbeitsdatei.

Bei einem SESAM-Sicherungsbestand auf Magnetbandkassette mit ARCHIVE sichert SESAM/SQL die PBI-Datei ebenfalls auf Magnetbandkassette. Das Einspielen der Datei erfolgt in diesem Fall erst bei der Reparatur oder beim Zurücksetzen.

Speichermedium, Primär- und Sekundärzuweisung

SESAM/SQL legt die PBI-Datei bzw. die HSMS-Arbeitsdatei als Datei nur für die Dauer der Online-Erstellung von SESAM-Sicherungsbeständen an. Das Anlegen erfolgt auf dem Speichermedium (Platte) und mit der Primär- und Sekundärzuweisung gemäß den Angaben der PBI-Mediensätze in der Medientabelle.

Für Primär- und Sekundärzuweisung der PBI-Dateibzw. die HSMS-Arbeitsdatei gelten die Standardwerte 576 (Primärzuweisung) und 72 (Sekundärzuweisung).

BS2000-Dateiname

Der BS2000-Dateiname für die PBI-Datei bzw. die HSMS-Arbeitsdatei lautet:
benutzerkennung.catalog.zeitstempel.01

benutzerkennung	BS2000-Benutzerkennung
catalog	Name der Datenbank
zeitstempel	Zeitpunkt der Kopieerstellung
01	Zähler

7.5.2.3 CAT-REC-Datei

Die CAT-REC-Datei (Catalog Recovery-Datei) dient als Steuerdatei für das Recovery des Catalog-Space und der gesamten Datenbank. Es gibt sie in verschiedenen Ausprägungen, wenn mit Logging-Dateien gearbeitet wird:

- als CAT-REC-Datei (Original);
  wird beim CREATE CATALOG erstellt oder beim ersten COPY der Datenbank, wenn die Datenbank ohne Logging-Dateien erstellt wurde.
  Sie beschreibt den aktuellen Stand des Originals.
- als CAT-REC-Kopie;
  wird beim COPY CATALOG[_SPACE] oder CHANGE-CATLOG angelegt als definierter Aufsetzpunkt für das Arbeiten mit Replikaten. Mit jedem COPY oder CHANGE-CATLOG wird eine eventuell vorhandene Kopie überschrieben.
  Auf diesen Stand kann dann ein Replikat nachgefahren werden.
- als CAT-REC-Datei des Replikats;
  wird beim CREATE REPLICATION angelegt. Sie enthält alle Sätze bis einschließlich des Sicherungssatzes, mit dem das Replikat erzeugt wurde.

  Benötigt wird diese Datei in zwei Situationen:

  – Wenn ein REFRESH REPLICATION durchgeführt werden soll. Aus der inhaltlichen Differenz der CAT-REC-Kopie und der CAT-REC-Datei des Replikats wird der Auftrag für REFRESH REPLICATION gebildet

  – Wenn mit RECOVER CATALOG USING/TO REPLICATION eine Originaldatenbank mit Hilfe des Replikats repariert, zurückgesetzt oder neu in das SQL-Datenbankverzeichnis eingetragen wird. Das Replikat der CAT-REC-Datei wird dabei zur CAT-REC-Datei des Originals.

Inhalt der CAT-REC-Datei

Die CAT-REC-Datei enthält Sätze mit folgenden Informationen:

- Kennblock
  enthält den Kennsatz und den Datenbanknamen.
- CREATE CATALOG-Satz
  enthält Angaben zum Catalog-Space aus Sicht des BS2000 wie z.B. Primär- und Sekundärzuweisung, BS2000-Kennwort und Speichermedium.
- Sätze zu den SESAM-Sicherungsbeständen des Catalog-Space
  Jeder Satz enthält Informationen zu einem bestimmten SESAM-Sicherungsbestand des Catalog-Space, so u.a. den Zeitstempel der Sicherung, die Versionsnummer des SESAM-Sicherungsbestandes, den Namen des SESAM-Sicherungsbestandes und das Speichermedium für den SESAM-Sicherungsbestand.
- Sätze zu den CAT-LOG-Dateien
  Jeder Satz enthält Informationen zu einer bestimmten CAT-LOG-Datei, so u.a. das Erstellungsdatum, die Versionsnummer des zugehörigen SESAM-Sicherungsbestandes des Catalog-Space sowie eine laufende Nummer für die CAT-LOG-Datei, die angibt, um die wievielte CAT-LOG-Datei für diesen SESAM-Sicherungsbestand es sich handelt.

Bei jeder Sicherung des Catalog-Space oder der Datenbank (siehe Seite 368) nimmt SESAM/SQL einen Satz für den zugehörigen SESAM-Sicherungsbestand des Catalog-Space in die CAT-REC-Datei auf.
Bei jedem Wechsel der CAT-LOG-Datei (siehe Abschnitt „CAT-LOG-Dateien“ auf Seite 218) nimmt SESAM/SQL einen entsprechenden Satz zu dieser CAT-LOG-Datei in die CAT-REC-Datei auf.

Mit Hilfe des Utility-Monitors kann sich der Datenbankverwalter über den Inhalt der CAT-REC-Datei informieren. Er kann auch die CAT-REC-Datei online oder offline pflegen (siehe Handbuch „Utility-Monitor“).

Mit der Utility-Anweisung MODIFY kann der Datenbankverwalter die CAT-REC-Datei online pflegen (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“).

Speichermedium, Primär- und Sekundärzuweisung

Bei einer Datenbank, die mit Logging betrieben wird (siehe Seite 226), legt SESAM/SQL die CAT-REC-Datei bei Ausführung der Utility-Anweisung CREATE CATALOG an. Bei Datenbanken ohne Logging erstellt SESAM/SQL die CAT-REC-Datei beim Erstellen des ersten SESAM-Sicherungsbestandes für die gesamte Datenbank oder den Catalog-Space. Das Anlegen erfolgt auf dem Speichermedium (Platte) und mit einer Primär- und Sekundärzuweisung gemäß den Angaben des ersten CAT-REC-Mediensatzes in der Medientabelle.

Für Primär- und Sekundärzuweisung der CAT-REC-Datei gilt jeweils der Standardwert 12.

BS2000-Dateinamen

Die BS2000-Dateinamen für die verschiedenen CAT-REC-Dateien lauten:
*benutzerkennung.catalog.*CAT-REC
*benutzerkennung.catalog.*CAT-REC.COPY
*benutzerkennung.replikat.*CAT-REC.REPL

benutzerkennung	BS2000-Benutzerkennung
catalog	Name der Datenbank
replikat	Name eines Replikats

Sichern der CAT-REC-Datei der Originaldatenbank

Die CAT-REC-Datei sollte mit DRV (Dual Recording by Volume) oder mit DUALCOPY doppelt geführt werden.

Eine Kopie der CAT-REC-Datei wird erzeugt durch COPY CATALOG[_SPACE] oder CHANGE-CATLOG. Mit jedem COPY oder CHANGE-CATLOG wird eine eventuell vorhandene Kopie überschrieben.
Die Originaldatenbank kann im RECOVER-Fall auf den Stand der CAT-REC-Kopie zurückgesetzt werden.

Bei COPY CATALOG[_SPACE] auf Magnetbandkassette mit HSMS kann in der HSMS-Parameterdatei angegeben werden, ob die CAT-REC-Datei mit gesichert werden soll.

7.5.2.4 CAT-LOG-Dateien

CAT-LOG-Dateien sind Logging-Dateien für den Catalog-Space (siehe Abschnitt „Logging“ auf Seite 226).

Inhalt der CAT-LOG-Datei

Auf CAT-LOG-Dateien protokolliert SESAM/SQL alle Änderungen im Catalog-Space. Änderungen im Catalog-Space erfolgen auf Grund von SQL-Anweisungen zur Verwaltung der Speicherstruktur, auf Grund von Utility-Anweisungen, von SQL-Anweisungen zur Verwaltung von Benutzereinträgen sowie auf Grund von SQL-Anweisungen zur Schemadefinition und Schemaverwaltung. Insbesondere protokolliert SESAM/SQL in CAT-LOG-Dateien alle Änderungen der Catalog-Tabellen RECOVERY_UNITS und DA_LOGS.
Außerdem dokumentiert SESAM/SQL in der CAT-LOG-Datei jede Kopieerstellung für die gesamte Datenbank sowie jede Kopieerstellung für den Catalog-Space. Eine Kopieerstellung für die Datenbank bzw. für den Catalog-Space bewirkt automatisch einen Wechsel der CAT-LOG-Datei. Die SESAM-Sicherungsbestände dienen als Aufsetzpunkte für das Nachfahren der protokollierten Änderungen bei der Reparatur der Datenbank bzw. des Catalog-Space (siehe Abschnitt „Datenbank, Catalog-Space und Anwender-Spaces wiederherstellen“ auf Seite 231).

Speichermedium, Primär- und Sekundärzuweisung

SESAM/SQL legt die erste CAT-LOG-Datei bei Ausführung der Utility-Anweisung CREATE CATALOG an. Das Anlegen erfolgt auf dem Speichermedium (Platte) und mit einer Primärzuweisung (Sekundärzuweisung ist stets 0) gemäß den Angaben des ersten CAT-LOG-Mediensatzes in der Medientabelle. Weitere CAT-LOG-Dateien legt SESAM/SQL ebenfalls entsprechend den Angaben in der Medientabelle an.

Für Primär- und Sekundärzuweisung der CAT-LOG-Dateien gelten die Standardwerte 768 (Primärzuweisung) und 0 (Sekundärzuweisung).

BS2000-Dateiname

Der BS2000-Dateiname für die CAT-LOG-Dateien lautet:
benutzerkennung.catalog.version.C.*nnnn*

benutzerkennung	BS2000-Benutzerkennung
catalog	Name der Datenbank
version	sechsstellige Versionsnummer eines SESAM-Sicherungsbestandes des Catalog-Space
nnnn	gibt an, um die wievielte CAT-LOG-Datei seit dem Erstellen des SESAM-Sicherungsbestandes mit der Nummer *version* für den Catalog-Space es sich handelt. Es können Lücken in der fortlaufenden Nummerierung entstehen.

Eröffnen der CAT-LOG-Datei

Wenn beim Eröffnen der CAT-LOG-Datei zum Eröffnungszeitpunkt die mit CREATE CATALOG oder ALTER MEDIA DESCRIPTION festgelegten Speichermedien für die CAT-LOG-Datei erschöpft bzw. nicht zugreifbar sind, tritt ein CC auf. Die Datenbank ist nach diesem CC nur noch lesend zugreifbar. Der CC hat den Eintrag im SQL-Datenbankverzeichnis von ACCESS=WRITE auf ACCESS=READ modifiziert.

Um ein ordnungsgemäßes Eröffnen zu ermöglichen, muss der Anwender auf dem betreffenden Medium Platz schaffen bzw. das Medium zugreifbar machen. Falls dies nicht möglich ist, muss ein CREATE-FILE-Kommando mit dem Namen der CAT-LOG-Datei (Name aus der Fehlermeldung in der SYSLST-Datei) eingegeben werden. Der Anwender muss mit dem Administrationsprogamm SESADM den Eintrag im SQL-Datenbankverzeichnis wieder auf ACCESS=WRITE setzen.

Wechsel der CAT-LOG-Datei

Einen Wechsel der CAT-LOG-Datei nimmt SESAM/SQL in folgenden Fällen vor:

- Ein neuer SESAM-Sicherungsbestand der gesamten Datenbank oder des Catalog-Space wurde erstellt. In diesem Fall erhält die neue CAT-LOG-Datei die Versionsnummer des neu erstellten SESAM-Sicherungsbestandes des Catalog-Space und die laufende Nummer 0001, z.B. *catalog*.000017.C.0001.
- Die laufende Nummer der DA-LOG-Datei läuft über (siehe Seite 221).
  In diesem Fall erhält die neue CAT-LOG-Datei die um eins erhöhte Versionsnummer und die laufende Nummer 0001, z.B. *catalog*.000018.C.0001.
- In der CAT-LOG-Datei ist kein freier Platz mehr vorhanden.
- Ein Fehler im BS2000-DMS ist aufgetreten.

In den beiden letztgenannten Fällen erhält die neue CAT-LOG-Datei dieselbe Versionsnummer wie die Vorgänger-Datei, die laufende Nummer wird um eins erhöht, z.B. *catalog*.000017.C.0002.

Zusätzlich können mit der Administrationsanweisung CHANGE-CATLOG die CAT-LOG-Datei und DA-LOG-Dateien zu der oder den angegebenen Datenbanken gewechselt werden (siehe Handbuch „Datenbankbetrieb“). Auch hier bleibt die Versionsnummer und die laufende Nummer erhöht sich um 1.

7.5.2.5 DA-LOG-Dateien

DA-LOG-Dateien sind Logging-Dateien für Anwender-Spaces (siehe Abschnitt „Logging“ auf Seite 226). Diese Logging-Dateien können mit dem Dienstprogramm SEDI70 ausgegeben werden (siehe Handbuch „Datenbankbetrieb“).

Inhalt der DA-LOG-Datei

Auf DA-LOG-Dateien protokolliert der SESAM/SQL-DBH alle Änderungen in Anwender-Spaces. Änderungen in Anwender-Spaces erfolgen auf Grund von SQL-Anweisungen zum Ändern von Daten sowie auf Grund von CALL-DML-Anweisungen.
Außerdem dokumentiert SESAM/SQL in der DA-LOG-Datei jede Kopieerstellung für die gesamte Datenbank bzw. den Catalog-Space sowie jede Kopieerstellung für einen Anwender-Space. Eine Kopieerstellung für die Datenbank bzw. den Catalog-Space bewirkt automatisch einen Wechsel der DA-LOG-Datei. Die SESAM-Sicherungsbestände dienen als Aufsetzpunkte für das Nachfahren der protokollierten Änderungen bei der Reparatur der Datenbank bzw. von Anwender-Spaces (siehe Abschnitt „Datenbank, Catalog-Space und Anwender-Spaces wiederherstellen“ auf Seite 231).

Speichermedium, Primär- und Sekundärzuweisung

SESAM/SQL legt die erste DA-LOG-Datei bei der ersten Änderung eines Anwender-Space an. Das Anlegen erfolgt auf dem Speichermedium (Platte) und mit der Primärzuweisung (Sekundärzuweisung ist stets 0) gemäß den Angaben des ersten DA-LOG-Mediensatzes in der Medientabelle. Weitere DA-LOG-Dateien legt SESAM/SQL ebenfalls entsprechend den Angaben in der Medientabelle an.

Für Primär- und Sekundärzuweisung der DA-LOG-Dateien gelten die Standardwerte 768 (Primärzuweisung) und 0 (Sekundärzuweisung).

BS2000-Dateiname

Der BS2000-Dateiname für die DA-LOG-Dateien lautet:
benutzerkennung.catalog.version.D.*nnnn*

benutzerkennung	BS2000-Benutzerkennung
catalog	Name der Datenbank
version	sechsstellige Versionsnummer eines SESAM-Sicherungsbestandes des Catalog-Space
nnnn	gibt an, um die wievielte DA-LOG-Datei seit dem Erstellen des SESAM-Sicherungsbestandes mit der Nummer *version* für den Catalog-Space es sich handelt. Es können Lücken in der fortlaufenden Nummerierung entstehen.

Eröffnen der DA-LOG-Datei

Wenn beim Eröffnen der DA-LOG-Datei zum Eröffnungszeitpunkt die mit CREATE MEDIA DESCRIPTION oder ALTER MEDIA DESCRIPTION festgelegten Speichermedien für die DA-LOG-Datei erschöpft bzw. nicht zugreifbar sind, tritt ein CC auf. Die Datenbank ist nach diesem CC nur noch lesend zugreifbar. Der CC hat den Eintrag im SQL-Datenbankverzeichnis von ACCESS=WRITE auf ACCESS=READ modifiziert.

Um ein ordnungsgemäßes Eröffnen zu ermöglichen, muss der Anwender auf dem betreffenden Medium Platz schaffen bzw. das Medium zugreifbar machen. Falls dies nicht möglich ist, muss ein CREATE-FILE-Kommando mit dem Namen der DA-LOG-Datei (Name aus der Fehlermeldung in der SYSLST-Datei) eingegeben werden. Anschließend muss er mit dem Administrationsprogamm SESADM den Eintrag im SQL-Datenbankverzeichnis wieder auf WRITE setzen.

Wechsel der DA-LOG-Datei

Einen Wechsel der DA-LOG-Datei nimmt SESAM/SQL in folgenden Fällen vor:

- Ein neuer SESAM-Sicherungsbestand der gesamten Datenbank oder des Catalog-Space wurde erstellt. In diesem Fall erhält die neue DA-LOG-Datei die Versionsnummer des neu erstellten SESAM-Sicherungsbestandes des Catalog-Space und die laufende Nummer 0001, z.B. *catalog*.000017.D.0001.
- In der DA-LOG-Datei ist kein freier Platz mehr vorhanden.
- Ein Fehler im BS2000-DMS ist aufgetreten.
- Ein Anwender-Space wurde repariert.
- Falls das Speichermedium eine Magnetbandkassette ist, erfolgt ein Wechsel der DA-LOG-Datei außerdem beim Starten des DBH sowie beim DBH-Wiederanlauf.

In den vier letztgenannten Fällen erhält die neue DA-LOG-Datei dieselbe Versionsnummer wie die Vorgänger-Datei, die laufende Nummer wird um eins erhöht, z.B. *catalog*.000017.D.0002.

Wenn die laufende Nummer *nnnn* der DA-LOG-Datei den Wert 9999 erreicht hat und SESAM/SQL die DA-LOG-Datei wechselt, dann erhält die neue DA-LOG-Datei die um eins erhöhte Versionsnummer und die laufende Nummer 0001, z.B. *catalog*.000018.D.0001. Zusätzlich wird eine neue CAT-LOG-Datei mit der um eins erhöhten Versionsnummer und der laufenden Nummer 0001 angelegt, z.B. *catalog*.000018.C.0001

Zusätzlich können mit der Administrationsanweisung CHANGE-DALOG die DA-LOG-Dateien zu der oder den angegebenen Datenbanken gewechselt werden (siehe Handbuch „Datenbankbetrieb“). Auch hier bleibt die Versionsnummer gleich und die laufende Nummer erhöht sich um 1.

7.5.2.6 Catalog-Tabelle RECOVERY_UNITS

Die Tabelle RECOVERY_UNITS ist eine von SESAM/SQL auf dem Catalog-Space der Datenbank angelegte Basistabelle. RECOVERY_UNITS dient zur Verwaltung der SESAM-Sicherungsbestände von Anwender-Spaces.

Inhalt von RECOVERY_UNITS

Jeder Satz von RECOVERY_UNITS enthält Informationen zu einem SESAM-Sicherungsbestand eines Anwender-Space, so u.a. den Namen des Anwender-Space, den dem SESAM-Sicherungsbestand zugeordneten Zeitstempel, sowie Informationen über die DA-LOG-Datei, die an den SESAM-Sicherungsbestand anschließt.
Jedesmal, wenn der Datenbankverwalter mit der Utility-Anweisung COPY einen SESAM-Sicherungsbestand eines Anwender-Space erstellt, nimmt SESAM/SQL einen entsprechenden Satz in die Tabelle RECOVERY_UNITS auf. Die Pflege der Tabelle RECOVERY_UNITS ist Aufgabe des Datenbankverwalters (siehe Abschnitt „Metadaten von SESAM-Sicherungsbeständen pflegen“ auf Seite 375).

Mit Hilfe des Utility-Monitors kann sich der Datenbankverwalter über den Inhalt der Catalog-Tabelle RECOVERY_UNITS informieren (siehe Handbuch „Utility-Monitor“).
Mit der Utility-Anweisung MODIFY kann der Datenbankverwalter die Catalog-Tabelle RECOVERY_UNITS pflegen (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“).

7.5.2.7 Catalog-Tabelle DA_LOGS

Die Tabelle DA_LOGS ist eine von SESAM/SQL auf dem Catalog-Space der Datenbank angelegte Basistabelle. DA_LOGS dient zur Verwaltung der DA-LOG-Dateien.

Inhalt von DA_LOGS

Jeder Satz von DA_LOGS enthält u.a. den Namen einer DA-LOG-Datei, den dieser DA-LOG-Datei zugeordneten Zeitstempel sowie Angaben, für welche Anwender-Spaces Änderungen in dieser DA-LOG-Datei protokolliert sind. In Verbindung mit der Catalog-Tabelle RECOVERY_UNITS liefert die Catalog-Tabelle DA_LOGS alle Informationen, die SESAM/SQL für die Reparatur von Anwender-Spaces benötigt.
Bei jedem Wechsel der DA-LOG-Datei nimmt SESAM/SQL einen entsprechenden Satz in die Catalog-Tabelle DA_LOGS auf. Die Pflege der Tabelle DA_LOGS ist Aufgabe des Datenbankverwalters (siehe Abschnitt „Metadaten von SESAM-Sicherungsbeständen pflegen“ auf Seite 375).

Mit Hilfe des Utility-Monitors kann sich der Datenbankverwalter über den Inhalt der Catalog-Tabelle DA_LOGS informieren (siehe Handbuch „Utility-Monitor“).
Mit der Utility-Anweisung MODIFY kann der Datenbankverwalter die Catalog-Tabelle DA_LOGS pflegen (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“).

7.5.3 SESAM-Sicherungsbestände erstellen

SESAM-Sicherungsbestände erstellt der Datenbankverwalter mit der Utility-Anweisung COPY (siehe Seite 368). Er kann die Sicherungsbestände auf Platte oder mit ARCHIVE bzw. HSMS auf Magnetbandkassette anlegen. Dabei ist darauf zu achten, dass die SESAM-Sicherungsbestände aus Sicherheitsgründen nicht auf derselben Platte liegen sollten wie der Catalog-Space und die Anwender-Spaces der Datenbank.

Sicherungseinheit festlegen

Als mögliche Einheiten für einen SESAM-Sicherungsbestand kommen infrage:

- gesamte SESAM/SQL-Datenbank.
  In diesem Fall werden der Catalog-Space und alle Anwender-Spaces der Datenbank gleichzeitig gesichert.
- Catalog-Space der Datenbank
- einzelner Anwender-Space
- Space-Set (Sicherungseinheit, bestehend aus mehreren Anwender-Spaces)

Ein Space-Set wird identifiziert durch den Zeitstempel, der in der Catalog-Tabelle RECOVERY_UNITS bei jedem Eintrag für einen Anwender-Space vermerkt ist. Der Zeitstempel gibt den Zeitpunkt der Kopieerstellung an, der für alle in einem Space-Set zusammengefassten Anwender-Spaces gleich sein muss. Spaces mit gleichem Zeitstempel entstehen bei der Sicherung mit einer COPY CATALOG-Anweisung oder bei der Sicherung mehrerer Spaces mit einer gemeinsamen COPY-Anweisung.

Bei der Wahl der Sicherungseinheit wird der Datenbankverwalter folgende Gesichtspunkte berücksichtigen:

- Dauer der Kopieerstellung.
  Die Dauer der Kopieerstellung richtet sich nach dem zu sichernden Datenvolumen.
- Dauer der Reparatur bzw. des Zurücksetzens.
  Die Dauer der Reparatur hängt ab vom Datenvolumen des einzuspielenden SESAM-Sicherungsbestandes sowie vom Änderungsvolumen, das seit Erstellung der Kopie angefallen ist. Die Dauer des Zurücksetzens hängt nur vom Datenvolumen des einzuspielenden SESAM-Sicherungsbestandes ab.
- Bei der Wahl des Space-Sets ist darauf zu achten, dass keine von SESAM/SQL geprüften Beziehungen von Anwender-Spaces dieses Space-Sets zu Anwender-Spaces außerhalb des Space-Sets bestehen (z.B. Referentielle Integrität, Indizes). Entsprechendes gilt für einen einzelnen Anwender-Space.
- Platzbedarf für die SESAM-Sicherungsbestände.
  Der Platzbedarf hängt ab vom zu sichernden Datenvolumen.

- Bei COPY CATALOG können Spaces von der Sicherung ausgenommen werden, falls für sie die beiden folgenden Bedingungen gleichzeitig gelten:
  - Die Spaces sind reine Index-Spaces und
  - die logische Datensicherung für diese Spaces ist ausgeschaltet.

  Bei RECOVER CATALOG .. GENERATE INDEX ON NO LOG INDEX SPACE werden für diese Spaces keine Sicherungen eingespielt. Statt dessen werden sie zurückgesetzt und neu aufgebaut.

Zeitpunkt für das Erstellen von SESAM-Sicherungsbeständen festlegen

Die Entscheidung, wann SESAM-Sicherungsbestände zu erstellen sind, wird bestimmt durch

- den auf die Datenbankanwendung zugeschnittenen Sicherungsplan des Datenbankverwalters.
- die Ausführung bestimmter Wartungsfunktionen für die Datenbank, bei denen eine Kopieerstellung sinnvoll und in einigen Fällen sogar notwendig ist.

SESAM-Sicherungsbestände gemäß Sicherungsplan erstellen

In der Regel wird der Datenbankverwalter jeweils periodisch SESAM-Sicherungsbestände der gesamten Datenbank, des Catalog-Space, von Space-Sets und von einzelnen Anwender-Spaces erstellen.

Für die zeitlichen Abstände, die zwischen zwei aufeinanderfolgenden Sicherungen derselben Sicherungseinheit liegen, kann sich der Datenbankverwalter z.B. an der Dauer der Kopieerstellung orientieren, die ihrerseits vom Datenvolumen der Sicherungseinheit abhängt. So ist es beispielsweise denkbar, Sicherungen der kompletten Datenbank monatlich durchzuführen, während Space-Sets und einzelne Anwender-Spaces wöchentlich gesichert werden.

Weitere Einflussfaktoren auf die Sicherungsperiode sind die Häufigkeit der vorgenommenen Änderungen sowie die Anforderungen an eine schnelle Verfügbarkeit nach einer Fehlersituation, d.h. kurze Reparaturzeiten. Von häufig geänderten Anwender-Spaces mit geforderten kurzen Reparaturzeiten wird der Datenbankverwalter daher öfter SESAM-Sicherungsbestände erstellen als von anderen Anwender-Spaces.

SESAM-Sicherungsbestände im Zusammenhang mit Wartungsfunktionen erstellen

Im Zusammenhang mit der Ausführung bestimmter Wartungsfunktionen für die Datenbank ist es sinnvoll und teilweise sogar notwendig, SESAM-Sicherungsbestände der Datenbank bzw. der betroffenen Anwender-Spaces zu erstellen. Wann dies im Einzelnen der Fall ist, ist im Kapitel „Datenbank aufbauen, laden und warten“ auf Seite 339 in den entsprechenden Abschnitten beschrieben.

Im Anschluss an die Ausführung der Utility-Anweisung CREATE CATALOG sollte der Datenbankverwalter einen SESAM-Sicherungsbestand des Catalog-Space als ersten Aufsetzpunkt für eventuelle Reparaturmaßnahmen erstellen. Ebenso sollte nach Ausführung jeder CREATE SPACE-Anweisung ein SESAM-Sicherungsbestand des betreffenden Anwender-Space erstellt werden, da das Wiederherstellen eines Anwender-Space ohne zugehörigen SESAM-Sicherungsbestand nur im Rahmen einer Reparatur der gesamten Datenbank möglich ist.

BS2000-Dateiname für die SESAM-Sicherungsbestände

Die Namen für die SESAM-Sicherungsbestände lauten
- beim Catalog-Space: *benutzerkennung.catalog.*CATALOG.*version*
- bei den Anwender-Spaces: *benutzerkennung.catalog.space.version*
- bei der CAT-REC-Datei: *benutzerkennung.catalog.*CAT-REC.COPY

benutzerkennung	BS2000-Benutzerkennung
catalog	Name der Datenbank, deren Catalog-Space bzw. Anwender-Space gesichert wurde
space	Name des gesicherten Anwender-Space
version	Versionsnummer

Die Versionsnummer ist eine sechsstellige ganze Zahl. Sie lautet 000002 für den ersten SESAM-Sicherungsbestand des Catalog-Space. Die Versionsnummer wird für SESAM-Sicherungsbestände des Catalog-Space mit jeder weiteren Kopieerstellung für den Catalog-Space um 1 erhöht.
Die Versionsnummer für den ersten SESAM-Sicherungsbestand eines beliebigen Anwender-Space lautet 000001 und wird für jeden weiteren SESAM-Sicherungsbestand des betreffenden Anwender-Space um 1 erhöht. Es können Lücken in der fortlaufenden Nummerierung entstehen, wenn COPY-Anweisungen wegen Fehler abgewiesen werden.

Bei der Erstellung eines SESAM-Sicherungsbestandes der gesamten Datenbank oder eines Space-Sets können die dabei erzeugten SESAM-Sicherungsbestände von Anwender-Spaces verschiedene Versionsnummern erhalten. Jeder SESAM-Sicherungsbestand eines betroffenen Anwender-Space erhält die für ihn gültige nächsthöhere Versionsnummer.

Bei der Sicherung des Catalog-Space wird automatisch auch die CAT-REC-Datei gesichert. Der Name der CAT-REC-Kopie enthält keine Versionsnummer. Das bedeutet, dass die Datei bei jedem erneuten COPY CATALOG[_SPACE] überschrieben wird.

Lesezugriffe auf SESAM-Sicherungsbestände

Auf einen SESAM-Sicherungsbestand einer gesamten SESAM/SQL-Datenbank kann der Anwender unter einem separaten DBH lesend zugreifen. Dies entlastet den laufenden Betrieb mit der Originaldatenbank.

Der Anwender kann den SESAM-Sicherungsbestand mit der Administrationsanweisung ADD-SQL-DB-CATALOG-ENTRY in das SQL-Datenbankverzeichnis einfügen. Dazu muss er bei dem Operanden COPY-NUMBER die Nummer des SESAM-Sicherungsbestandes des Catalog-Space angeben.

Erlaubt sind nur lesende DML-Zugriffe, d.h. auch keine Utility-Anweisungen. Der SESAM-Sicherungsbestand, auf den lesend zugegriffen wird, steht unverändert für spätere Recovery-Maßnahmen zur Verfügung.

7.5.4 Fremdkopien und Replikate als Sicherungsbestände

Außer SESAM-Sicherungsbeständen können auch Fremdkopien (siehe Abschnitt „Fremdkopien einer Datenbank einsetzen“ auf Seite 382) und Replikate (siehe Abschnitt „Replikat einer Datenbank“ auf Seite 247) zur Media-Recovery verwendet werden.

7.5.5 Logging

SESAM/SQL protokolliert alle Änderungen in der Datenbank, die seit dem Erstellen eines bestimmten SESAM-Sicherungsbestandes angefallen sind, in Logging-Dateien (siehe Abschnitt „Dateien und Tabellen für das Media-Recovery“ auf Seite 212).

Falls die Datenbank auf einer DB-Kennung liegt und die Logging-Dateien ebenfalls auf dieser Kennung liegen sollen, so müssen dafür Vorbereitungen getroffen werden, siehe Abschnitt „Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen“ auf Seite 308.

Beim Logging ist zu unterscheiden:

- Änderungen, die den Catalog-Space der Datenbank betreffen, d.h. Änderungen auf Grund von Utility-Anweisungen, von SQL-Anweisungen zur Verwaltung der Speicherstruktur, zur Verwaltung von Benutzereinträgen sowie zur Schemadefinition und Schemaverwaltung, protokolliert SESAM/SQL auf CAT-LOG-Dateien (CAT-Logging).
- Änderungen in den Anwender-Spaces, d.h. Änderungen auf Grund von SQL-Anweisungen zum Ändern der Daten sowie auf Grund von CALL-DML-Anweisungen, protokolliert SESAM/SQL auf DA-LOG-Dateien (DA-Logging).

Logging für die Datenbank vereinbaren

Beim Anlegen der Datenbank mit der Utility-Anweisung CREATE CATALOG (siehe Seite 341) vereinbart der Datenbankverwalter, ob die Datenbank mit Logging betrieben werden soll. Das Logging erstreckt sich dann auf die gesamte Datenbank, d.h. auf den Catalog-Space sowie die zugehörigen Anwender-Spaces. Außerdem lässt sich das Logging durch Angabe von LOG bei COPY CATALOG einschalten!

Um Datenverlust durch System- und Plattenfehler zu vermeiden, sollte eine Datenbank in der Regel mit Logging betrieben werden. Datenbankbetrieb ohne Logging bietet sich an bei Testdatenbanken oder bei Datenbanken für temporäre Daten, wenn sich Datenverlust auf andere Weise vermeiden lässt, sowie bei Datenbanken, die vorwiegend oder ausschließlich zur Datenwiedergewinnung eingesetzt werden.

Logging für Anwender-Spaces vereinbaren

Sofern das Logging für die Datenbank vereinbart ist, wird es im Standardfall auch für jeden mit CREATE SPACE erzeugten Anwender-Space durchgeführt (siehe Seite 350). Der Datenbankverwalter kann jedoch einen Anwender-Space auch ohne Logging vereinbaren. Mit ALTER SPACE kann der Datenbankverwalter darüber hinaus bei einem Anwender-Space, für den Logging zunächst vereinbart war, das Logging ausschalten.
Wenn Logging für die Datenbank nicht vereinbart ist, ist Logging für die Anwender-Spaces dieser Datenbank ebenfalls nicht möglich.

Einen Anwender-Space ohne Logging zu vereinbaren, kann u.U. sinnvoll sein, wenn der Anwender-Space ausschließlich Indizes oder Tabellen mit temporären Daten enthält. Zerstörte Indizes kann der Datenbankverwalter im Fehlerfall mit RECOVER INDEX wiederherstellen (siehe Abschnitt „Indizes wiederherstellen“ auf Seite 246).

Bei COPY CATALOG können reine Index-Spaces, die nicht in der logischen Datensicherung sind, von der Sicherung mit COPY ausgenommen werden. Bei einem anschließenden RECOVER muss die Klausel GENERATE INDEX ON NO LOG INDEX SPACE angegeben werden. Für die Index-Spaces werden dann keine Sicherungen eingespielt. Statt dessen werden sie zurückgesetzt und neu aufgebaut.

Bei der Entscheidung, einen Anwender-Space, auf dem ausschließlich Indizes liegen, mit oder ohne Logging zu betreiben, sollte der Datenbankverwalter beachten: Beim Betrieb ohne Logging entsteht ein Performance-Gewinn. Andererseits erfordert die Neugenerierung fehlerhafter oder zerstörter Indizes mit RECOVER INDEX mehr Zeit als die Wiederherstellung der Indizes im Rahmen des allgemeinen Reparaturverfahrens. Welches Verfahren sich anbietet, ist somit anhand des konkreten Anwendungsfalls zu entscheiden.

Logging für Anwender-Spaces ausschalten bzw. unterbrechen

Jede Unterbrechung des Logging für einen Anwender-Space dokumentiert SESAM/SQL in der Catalog-Tabelle RECOVERY_UNITS durch einen Eintrag mit dem Zeitstempel der Unterbrechung.

Ausschalten bzw. Unterbrechen des Logging durch den Datenbankverwalter

Das Logging für einen Anwender-Space kann der Datenbankverwalter mit der SQL-Anweisung ALTER SPACE vorübergehend ausschalten (siehe Seite 350). Dieses Vorgehen bietet sich beispielsweise an, wenn ein größerer Änderungslauf beschleunigt durchgeführt werden soll.

Der Datenbankverwalter sollte dafür sorgen, dass im Fehlerfall der Zustand des Anwender-Space zum Zeitpunkt der Logging-Unterbrechung, also unmittelbar vor Beginn des Änderungslaufs, wiederhergestellt werden kann. Zu Reparaturzwecken genügt es dann, den Änderungslauf zu wiederholen.
Der Datenbankverwalter erreicht dies wie folgt:

- Falls es sich um die erste Logging-Unterbrechung seit Erstellung des bislang letzten SESAM-Sicherungsbestandes für den Anwender-Space handelt, stellt der Datenbankverwalter im Fehlerfall den Zustand des Anwender-Space zum Zeitpunkt der Logging-Unterbrechung wieder her mit der Utility-Anweisung RECOVER TO und Angabe des Zeitstempels der Logging-Unterbrechung.
- Falls es sich nicht um die erste Logging-Unterbrechung seit Erstellung des bislang letzten SESAM-Sicherungsbestandes für den Anwender-Space handelt, muss der Datenbankverwalter unmittelbar vor Ausschalten des Logging einen SESAM-Sicherungsbestand des Anwender-Space mit der Utility-Anweisung COPY erstellen (siehe Seite 368). Auf diesen SESAM-Sicherungsbestand kann im Fehlerfall mit der Utility-Anweisung RECOVER TO zurückgesetzt werden.

Nach dem Änderungslauf sollte der Datenbankverwalter erneut einen SESAM-Sicherungsbestand des Anwender-Space mit der Utility-Anweisung COPY erstellen und mit COPY gleichzeitig das Logging für diesen Anwender-Space wieder einschalten. Dieser SESAM-Sicherungsbestand bildet dann den Ausgangspunkt für weitere Recovery-Maßnahmen auf diesem Anwender-Space.

Statt eines SESAM-Sicherungsbestandes kann auch eine Fremdkopie erzeugt und das Logging mit PREPARE-FOREIGN-COPY eingeschaltet werden, siehe Seite 382 und das Handbuch „Datenbankbetrieb“.

Unterbrechen des Logging durch SESAM/SQL bei einigen Utility-Anweisungen

Automatisch unterbricht SESAM/SQL das Logging für die betroffenen Anwender-Spaces bei Ausführung der Utility-Anweisungen LOAD OFFLINE, IMPORT TABLE, RECOVER INDEX, RECOVER ADJUST, REORG ... NEW ROW_IDS sowie bei MIGRATE, wenn es sich nicht um einen MIGRATE CALL DML ONLY TABLE handelt. Unmittelbar vor Ausführung dieser Utility-Anweisungen sollte der Datenbankverwalter daher SESAM-Sicherungsbestände der betroffenen Anwender-Spaces erstellen. Tritt während der Ausführung von LOAD OFFLINE, IMPORT TABLE, MIGRATE, RECOVER INDEX, RECOVER ADJUST bzw. REORG ... NEW ROW_IDS ein Fehler auf, kann so auf den Stand unmittelbar vor Ausführung der betreffenden Utility-Anweisung zurückgesetzt und die Anweisung wiederholt werden. Bei einem RECOVER nach einer fehlerhaften Ausführung von IMPORT TABLE wird die Tabelle unter Umständen angelegt. Sie muss in diesem Fall mit DROP TABLE gelöscht werden, bevor IMPORT TABLE wiederholt werden kann. Bei einer fehlerhaften Ausführung von MIGRATE ist zu beachten, dass der Datenbankverwalter eine vom fehlerhaften MIGRATE bereits erzeugte Tabelle vor einer erneuten Ausführung von MIGRATE manuell mit der SQL-Anweisung DROP TABLE löschen muss.

Unmittelbar nach Ausführung von LOAD OFFLINE, IMPORT TABLE, MIGRATE, RECOVER INDEX, RECOVER ADJUST und REORG ... NEW ROW_IDS muss der Datenbankverwalter erneut SESAM-Sicherungsbestände der betroffenen Anwender-Spaces erstellen, die als neuer Ausgangspunkt für das weitere Logging dienen. Andernfalls befinden sich diese Anwender-Spaces im Zustand „copy pending“ (siehe Seite 166). Sie bleiben für Anweisungen zum Ändern von Daten solange gesperrt, bis der Datenbankverwalter entsprechende SESAM-Sicherungsbestände erstellt hat.

Speichergeräte für das Logging festlegen

Die Rekonstruktion des Catalog-Space bzw. eines Anwender-Space nach Plattenfehler ist nur möglich, wenn die SESAM-Sicherungsbestände, die CAT-REC-Datei sowie die CAT-LOG- und DA-LOG-Dateien auf anderen Speichergeräten liegen als der Catalog-Space und die Anwender-Spaces.

Beim Anlegen des Catalog-Space mit der Utility-Anweisung CREATE CATALOG definiert der Datenbankverwalter die Storage Group und damit das Speichermedium, auf dem SESAM/SQL die CAT-REC-Datei und die CAT-LOG-Dateien anlegt (siehe Abschnitt „Catalog-Space der Datenbank anlegen“ auf Seite 341). Nach dem Erstellen des Catalog-Space kann der Datenbankverwalter mit der SQL-Anweisung CREATE STOGROUP weitere Storage Groups definieren (siehe Seite 348).

Die Speichermedien für die DA-LOG-Dateien sowie eventuell benötigte weitere Speichermedien für die CAT-LOG-Dateien ordnet der Datenbankverwalter über die Medientabelle zu. CAT-LOG- und DA-LOG-Dateien können nur auf Platte angelegt werden, wobei eine Platte durch die betreffende Storage Group identifiziert wird.

7.5.6 SESAM-Sicherungsbestände und Logging-Dateien verwalten

SESAM-Sicherungsbestände des Catalog-Space und CAT-LOG-Dateien verwalten

Für jeden SESAM-Sicherungsbestand des Catalog-Space gibt es in der CAT-REC-Datei einen entsprechenden Eintrag. Im Anschluss an jeden solchen Eintrag folgen die Einträge für die jeweils zugehörigen CAT-LOG-Dateien.

Der Datenbankverwalter kann die CAT-REC-Datei bearbeiten, d.h. Einträge löschen und auf diese Weise die SESAM-Sicherungsbestände des Catalog-Space verwalten. Dies geschieht mit der Utility-Anweisung MODIFY (online-update, siehe Seite 375) oder mit Hilfe des Utility-Monitors (offline-update, siehe Handbuch „Utility-Monitor“).
Nach Löschen eines Eintrags aus der CAT-REC-Datei ist der zugehörige SESAM-Sicherungsbestand bzw. die CAT-LOG-Datei für SESAM/SQL unbekannt und steht für Recovery-Zwecke nicht mehr zur Verfügung.

Es wird empfohlen, nicht mehr benötigte SESAM-Sicherungsbestände und Logging-Dateien im BS2000 zu löschen (siehe Seite 377).

SESAM-Sicherungsbestände von Anwender-Spaces und DA-LOG-Dateien verwalten

Für jeden SESAM-Sicherungsbestand eines Anwender-Space enthält die Catalog-Tabelle RECOVERY_UNITS einen entsprechenden Eintrag. Informationen über DA-LOG-Dateien sind in der Catalog-Tabelle DA_LOGS vermerkt.

Mit Hilfe der Utility-Anweisung MODIFY kann der Datenbankverwalter die Catalog-Tabellen RECOVERY_UNITS und DA_LOGS pflegen (siehe Seite 375). Nach dem Löschen eines Eintrags in RECOVERY_UNITS bzw. DA_LOGS ist der entsprechende SESAM-Sicherungsbestand bzw. die DA-LOG-Datei für SESAM/SQL unbekannt und steht für Recovery-Zwecke nicht mehr zur Verfügung.

Es wird empfohlen, nicht mehr benötigte SESAM-Sicherungsbestände und Logging-Dateien im BS2000 zu löschen (siehe Seite 377).

7.5.7 Datenbank, Catalog-Space und Anwender-Spaces wiederherstellen

Beim Wiederherstellen unterscheidet man zwischen Reparatur und Zurücksetzen. Reparatur und Zurücksetzen führt der Datenbankverwalter mit der Utility-Anweisung RECOVER durch (siehe Seite 377).
Entsprechend den möglichen Sicherungseinheiten kann der Datenbankverwalter zwischen folgenden Einheiten für Reparatur und Zurücksetzen wählen:

- gesamte SESAM/SQL-Datenbank mit Catalog-Space und allen Anwender-Spaces
- Catalog-Space der Datenbank
- einzelner Anwender-Space, Space-Liste oder Space-Set

Als Space-Set wird eine Einheit aus mehreren Anwender-Spaces mit gemeinsamem Zeitstempel bezeichnet. Spaces mit gleichem Zeitstempel entstehen bei der Sicherung mit einer COPY CATALOG-Anweisung oder bei der Sicherung mehrerer Spaces mit einer gemeinsamen COPY-Anweisung. Die Reparatur eines Space-Set als Einheit ist nur möglich, wenn außerdem das Logging zwischenzeitlich für keinen Anwender-Space dieses Space-Sets unterbrochen wurde.
Eine Space-Liste bezeichnet mehrere Anwender-Spaces aus der Sicherung einer gesamten Datenbank oder eines Space-Sets, die für eine gemeinsame RECOVER-Anweisung zusammengestellt werden. Die Spaces einer Space-Liste müssen alle den gleichen Zeitstempel tragen. Die Reparatur der Spaces einer Space-Liste ist nur möglich, wenn außerdem das Logging zwischenzeitlich für keinen Anwender-Space dieser Space-Liste unterbrochen wurde.

Zur Verkürzung der Recovery-Zeiten können Sie die Menge der Spaces bei RECOVER zusätzlich einschränken:

- Geben Sie bei RECOVER die Klausel SCOPE PENDING an, werden die Anwender-Spaces repariert, wenn sie beim Öffnen als defekt erkannt werden.
  SCOPE PENDING ist bei Fremdkopien sowie bei RECOVER SPACESET ... TO oder RECOVER SPACE *space-liste* TO nicht erlaubt.

  Bei RECOVER CATALOG wird der Catalog-Space immer repariert und die Anwender-Spaces nur, wenn sie beim Öffnen als defekt erkannt werden.

  Bei RECOVER CATALOG ... TO wird der Catalog-Space immer zurückgesetzt.Die Anwender-Spaces werden nur zurückgesetzt, wenn sie beim Öffnen als defekt erkannt werden oder wenn aus Sicht des Catalogs eine Inkonsistenz zwischen Catalog-Space und Anwender-Space festgestellt wird. Eine solche Inkonsistenz besteht, wenn der Anwender-Space seit dem Zeitpunkt der Sicherung, auf die zurückgesetzt wird, geändert wurde.

- Verwalten Sie Indizes auf Index-Spaces, d.h. auf Spaces, auf denen ausschließlich Indizes liegen und die nicht in der logischen Datensicherung sind. Zur Arbeit mit diesen Index-Spaces gibt es zwei Klauseln.
  - GENERATE INDEX ON NO LOG INDEX SPACE
    Diese Klausel können Sie bei RECOVER CATALOG angeben. Mit ihr werden die Spaces, auf denen ausschließlich Indizes liegen und die nicht in der logischen Datensicherung sind, zurückgesetzt und die Indizes neu aufgebaut.
  - NO INDEX
    Diese Klausel können Sie beim Zurücksetzen eines Anwender-Space, einer Space-Liste oder eines Space-Set angeben. Mit ihr werden Indizes als defekt gekennzeichnet und nicht neu aufgebaut, wenn sie durch das Zurücksetzen eines Space ungültig geworden sind. Dieser Fall tritt auf, falls die Basistabelle oder die partitionierte Tabelle und ein zugehöriger Index auf unterschiedlichen Anwender-Spaces liegen und nicht alle betroffenen Anwender-Spaces gleichzeitig zurückgesetzt werden.

Reparatur

Grundlage für die Reparatur bildet ein zuvor erstellter SESAM-Sicherungsbestand der Datenbank, des Catalog-Space oder eines Anwender-Space, einer Space-Liste bzw. eines Space-Sets und die daran anschließenden Logging-Dateien. Logging-Dateien für den Catalog-Space sind die CAT-LOG-Dateien, Logging-Dateien für die Anwender-Spaces sind die DA-LOG-Dateien.

Die zusammengehörigen Sicherungsdateien kann sich der Datenbankverwalter in Jobvariablen ausgeben lassen und zwar mit Hilfe der INF-Maske des Utility-Monitors. Über vorhandene SESAM-Sicherungsbestände des Catalog-Space informiert sich der Datenbankverwalter mit Hilfe des Utility-Monitors in der CAT-REC-Datei. Welche SESAM-Sicherungsbestände von den einzelnen Anwender-Spaces vorliegen, ermittelt der Datenbankverwalter ebenfalls mit Hilfe des Utility-Monitors aus der Catalog-Tabelle RECOVERY_UNITS (siehe dazu Handbuch „Utility-Monitor“).
Die Reparatur bringt die Datenbank, den Catalog-Space oder einen Anwender-Space, eine Space-Liste bzw. ein Space-Set auf den Stand, der vor Eintreten der Fehlersituation vorlag.

Anwender-Spaces, der Catalog-Space oder die gesamte Datenbank können auch mit Hilfe von Fremdkopien repariert werden (siehe Abschnitt „Reparieren und Zurücksetzen mit Fremdkopien“ auf Seite 392).

Die Reparatur von Anwender-Spaces, des Catalog-Space oder der gesamten Datenbank ist auch auf Basis eines Replikats möglich (siehe Abschnitt „Originaldatenbank mit einem Replikat reparieren“ auf Seite 256).

Reparatur des Catalog-Space

Bei der Reparatur des Catalog-Space entnimmt SESAM/SQL aus der CAT-REC-Datei die Information über den SESAM-Sicherungsbestand des Catalog-Space, den der Datenbankverwalter bei RECOVER angegeben hat, und die Information über die zugehörigen CAT-LOG-Dateien. Anschließend spielt SESAM/SQL diesen SESAM-Sicherungsbestand ein und fährt auf diesem SESAM-Sicherungsbestand die in den CAT-LOG-Dateien protokollierten Änderungen nach.

Bild 18 veranschaulicht die Reparatur des Catalog-Space.

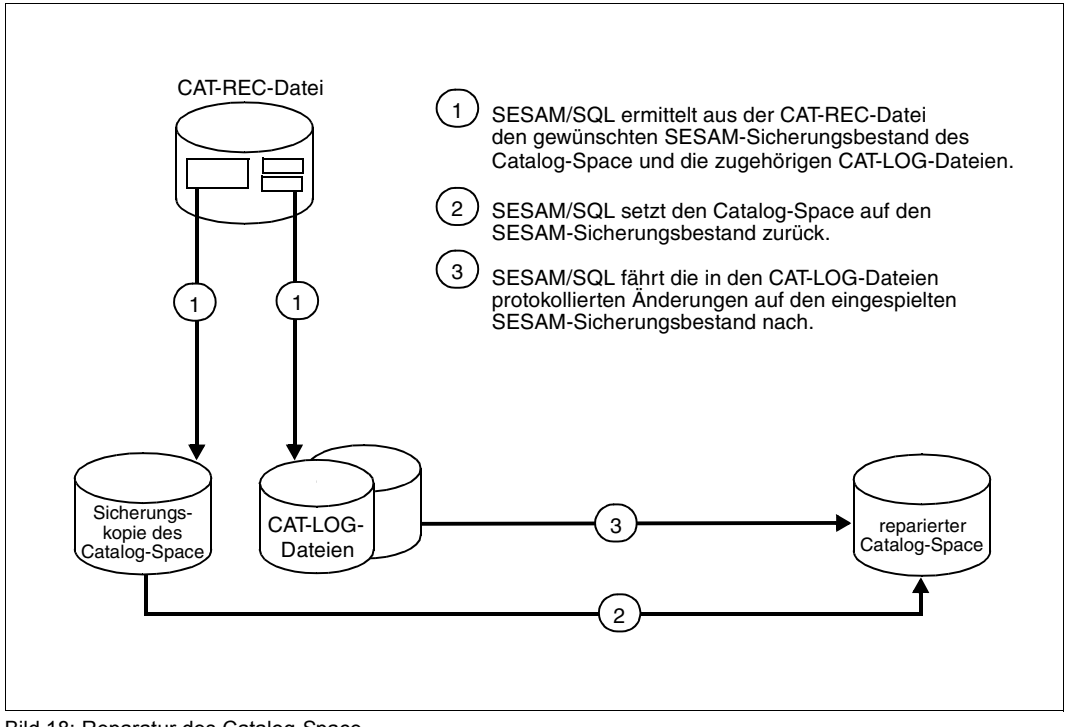


Bild 18: Reparatur des Catalog-Space

Reparatur von Anwender-Spaces

Bei der Reparatur eines Anwender-Space, einer Space-Liste oder eines Space-Set entnimmt SESAM/SQL aus der Catalog-Tabelle RECOVERY_UNITS Informationen über den SESAM-Sicherungsbestand jedes Anwender-Space, die der Datenbankverwalter bei RECOVER angegeben hat. Aus der Catalog-Tabelle DA_LOGS ermittelt SESAM/SQL die DA-LOG-Dateien, auf denen die Änderungen des Anwender-Space seit der Erstellung dieser Kopie protokolliert sind. Anschließend spielt SESAM/SQL den SESAM-Sicherungsbestand ein und fährt auf diesen SESAM-Sicherungsbestand die in den DA-LOG-Dateien protokollierten Änderungen nach.

Bild 19 veranschaulicht die Reparatur eines Anwender-Space.

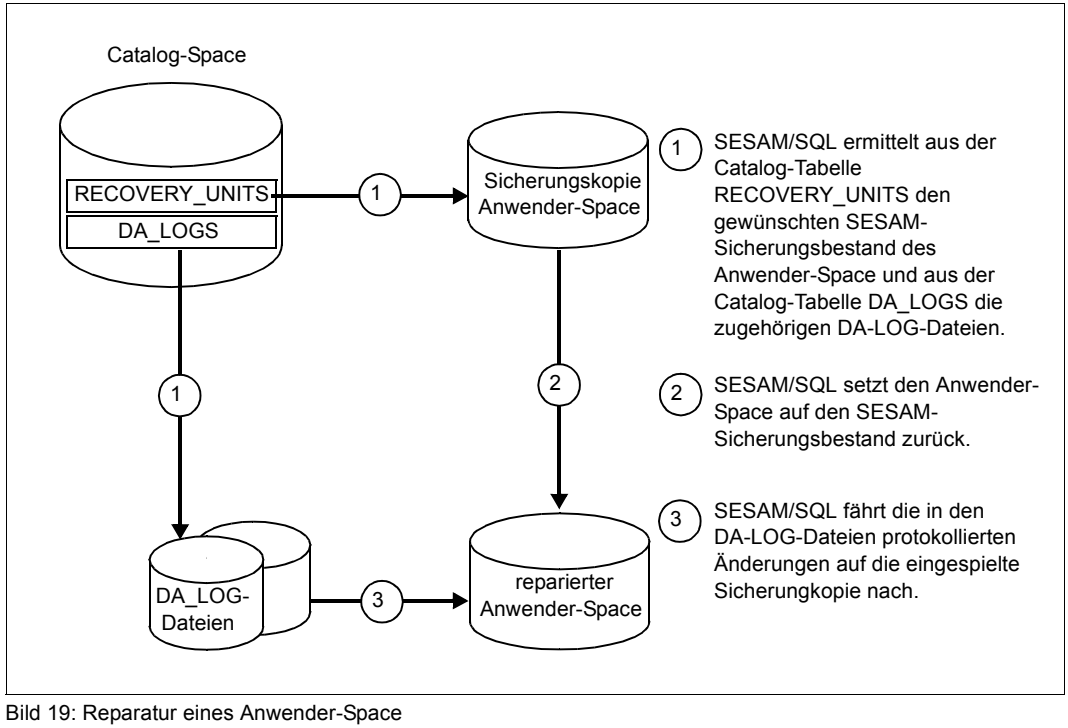


Bild 19: Reparatur eines Anwender-Space

Reparatur der gesamten SESAM/SQL-Datenbank

Bei der Reparatur der gesamten Datenbank führt SESAM/SQL zunächst die Reparatur des Catalog-Space auf der Basis des bei RECOVER angegebenen SESAM-Sicherungsbestandes durch. Mit Hilfe der Metadaten des wiederhergestellten Catalogs repariert SESAM/SQL anschließend alle Anwender-Spaces der Datenbank auf der Basis des jeweils zuletzt erstellten SESAM-Sicherungsbestandes der einzelnen Anwender-Spaces. Demzufolge ist die Konsistenz der Datenbank nach ordnungsgemäßer Reparatur stets gewährleistet.
Hat SESAM/SQL jedoch die Reparatur eines Anwender-Space z.B. wegen einer fehlenden oder defekten DA-LOG-Datei abgebrochen und den Anwender-Space im gerade vorliegenden Zustand „eingefroren“, dann kann der Datenbankverwalter wahlweise die Reparatur fortsetzen oder die Konsistenz der Datenbank in dem eingefrorenen Zustand herstellen.

Eine Reparatur, die SESAM/SQL wegen einer fehlenden oder defekten DA-LOG-Datei zunächst abgebrochen hat, kann der Datenbankverwalter mit RECOVER RESTART nach Bereitstellen einer entsprechenden intakten DA-LOG-Datei ordnungsgemäß zu Ende führen. Dazu muss der Datenbankverwalter die Utility-Anweisung RECOVER RESTART für jeden Anwender-Space ausführen.

Die Konsistenz der Datenbank in dem eingefrorenen Zustand stellt der Datenbankverwalter her, indem er die Utility-Anweisung RECOVER ADJUST für jeden Anwender-Space ausführt. Bei RECOVER ADJUST stellt SESAM/SQL die Konsistenz über dieselben Anpassungsmechanismen wieder her wie beim Zurücksetzen einzelner Anwender-Spaces (siehe „Konsistenz der Datenbank beim Zurücksetzen“ auf Seite 240).

Zurücksetzen

Zurücksetzen als Maßnahme zum Wiederherstellen einzelner Anwender-Spaces, einer Space-Liste, eines Space-Set, des Catalog-Space oder der gesamten Datenbank bietet sich immer dann an, wenn für die betreffende Sicherungseinheit das Logging ganz oder vorübergehend ausgeschaltet war (siehe Seite 226) und demzufolge eine Reparatur nicht möglich ist.

Ebenfalls nur durch Zurücksetzen auf die zuletzt erstellten SESAM-Sicherungsbestände der betroffenen Spaces lässt sich ein Fehler in der Datenbank beheben, der durch ein fehlerhaftes Anwenderprogramm verursacht wurde. Dies gilt auch, wenn das Logging eingeschaltet war. Ein selektives Nachfahren der DA-LOG-Dateien, das die Datenänderungen durch das fehlerhafte Anwenderprogramm ausblendet, ist nicht möglich.

Anwender-Spaces, der Catalog-Space oder die gesamte Datenbank können auch mit Hilfe von Fremdkopien zurückgesetzt werden (siehe Abschnitt „Reparieren und Zurücksetzen mit Fremdkopien“ auf Seite 392).

Das Zurücksetzen von Anwender-Spaces, des Catalog-Space oder der gesamten Datenbank ist auch auf Basis eines Replikats möglich (siehe Abschnitt „Originaldatenbank mit einem Replikat reparieren“ auf Seite 256).

Zurücksetzen von Anwender-Spaces

Einen Anwender-Space, eine Space-Liste oder einen Space-Set setzt der Datenbankverwalter mit der Utility-Anweisung RECOVER TO zurück.

Grundlage für das Zurücksetzen bildet ein zuvor erstellter SESAM-Sicherungsbestand eines Anwender-Space, einer Space-Liste oder eines Space-Sets. Bei der Angabe einer Space-Liste oder eines Space-Sets müssen alle angesprochenen Spaces den gleichen Sicherungs-Zeitstempel tragen. Das Zurücksetzen bringt den Anwender-Space bzw. die in der Space-Liste oder in dem Space-Set zusammengefassten Anwender-Spaces auf den Stand, der zum Zeitpunkt der Erstellung der betreffenden SESAM-Sicherungsbestände vorlag.

ACHTUNG!
Ein RECOVER TO kann nicht rückgängig gemacht werden.
Ein neuerer Stand kann nicht mehr erreicht werden.

Welche SESAM-Sicherungsbestände von den einzelnen Anwender-Spaces vorliegen, ermittelt der Datenbankverwalter mit Hilfe des Utility-Monitors aus der Catalog-Tabelle RECOVERY_UNITS.
Den gewünschten SESAM-Sicherungsbestand kann der Datenbankverwalter über die Versionsnummer der Sicherung, den Zeitstempel oder den Dateinamen des SESAM-Sicherungsbestandes auswählen.

Zurücksetzen von Anwender-Spaces auf eine Marke

Eine Marke ist ein spezieller Eintrag in der Catalog-Tabelle RECOVERY_UNITS des Informationsschemas. Sie besitzt einen Zeitstempel und erhält als RECOVER_TYPE die Bezeichnung „MARK“. Sie repräsentiert einen festgeschriebenen Stand der Datenbank, der auf Basis einer davor liegenden Sicherung über Nachfahren der Loggingdateien wiederhergestellt werden kann.

Eine Marke wird bei Ausführung der Utility-Funktionen LOAD OFFLINE, MIGRATE, IMPORT, RECOVER INDEX oder REORG ... NEW ROW_IDS bzw. bei der SSL-Anweisung ALTER SPACE ... NO LOG geschrieben. Mit Hilfe dieser Daten kann der Datenbankverwalter einen Anwender-Space mit der Utility-Anweisung RECOVER SPACE *space* USING *rec_unit* TO TIMESTAMP oder im Utility-Monitor über die Maske COP auf eine Marke zurücksetzen. *rec_unit* kann dabei ein SESAM-Sicherungsbestand, eine Fremdkopie oder ein Replikat sein.

Beispiel

Ein LOAD OFFLINE für den Space *space* soll rückgängig gemacht werden.

Folgende Arbeitsschritte sind nötig:

1. Sicherung bereitstellen.
2. Ermitteln Sie in der Catalog-Tabelle RECOVERY_UNITS des Informationsschemas den Eintrag, der den RECOVER_TYPE „MARK“ und den RECOVERY_TIMESTAMP *before_load* mit dem Zeitstempel des LOAD enthält.
3. RECOVER-Anweisung ausführen:
   `RECOVER SPACE` *space* `USING` *rec_unit* `TO TIMESTAMP` *before_load*

Wenn die Funktionen LOAD OFFLINE, MIGRATE, IMPORT, RECOVER INDEX oder REORG ... NEW ROW_IDS mit Fehler abgebrochen werden, dann ist normalerweise die Marke noch nicht geschrieben. Der Space kann dann mit `RECOVER SPACE` *space* `USING` auf den Stand vor dem Funktionsaufruf zurückgesetzt werden. Bei LOAD OFFLINE ist zu beachten, dass zum Zeitpunkt des Abbruchs bereits Indizes als defekt gekennzeichnet sein können. Diese Indizes werden durch RECOVER SPACE *space* USING nicht neu aufgebaut, so dass ggf. noch RECOVER INDEX erforderlich ist.

Zurücksetzen der gesamten SESAM/SQL-Datenbank

Die gesamte Datenbank setzt der Datenbankverwalter mit der Utility-Anweisung RECOVER CATALOG zurück.

Dabei gibt es zwei Möglichkeiten:

- Zurücksetzen auf einen zuvor erstellten SESAM-Sicherungsbestand mit RECOVER CATALOG TO:

  Das Zurücksetzen bringt die Datenbank auf den Stand, der zum Zeitpunkt der Erstellung des SESAM-Sicherungsbestandes vorlag.

  In diesem Fall führt SESAM/SQL zunächst das Zurücksetzen des Catalog-Space auf der Basis des bei RECOVER CATALOG TO angegebenen SESAM-Sicherungsbestandes durch. Mit Hilfe der Metadaten des zurückgesetzten Catalog-Space werden alle Anwender-Spaces der Datenbank auf der Basis des jeweils zuletzt erstellten SESAM-Sicherungsbestandes der einzelnen Anwender-Spaces repariert. Damit ist die Konsistenz der Datenbank nach ordnungsgemäßer Reparatur stets gewährleistet.

- Zurücksetzen auf einen (frei) wählbaren Zeitpunkt mit RECOVER CATALOG [USING ...] TO [ANY] *zeitstempel*:

  Das Zurücksetzen bringt die Datenbank auf den Stand, der zum angegebenen Zeitpunkt vorlag. Wenn ANY nicht angegeben wird, dann muss *zeitstempel* den Zeitpunkt eines SESAM-Sicherungsbestandes des Catalog-Space bezeichnen. Der SESAM-Sicherungsbestand muss in der CAT-REC-Datei eingetragen sein.

  SESAM/SQL spielt den den angegebenen SESAM-Sicherungsbestand des Catalog-Space (USING angegeben) oder den letzten SESAM-Sicherungsbestand des Catalog-Space vor dem angegebenen Zeitpunkt ein. Anschließend fährt SESAM/SQL die CAT-LOG-Dateien, die an diesen SESAM-Sicherungsbestand anschließen, bis zum angegebenen Zeitpunkt nach.

  Mit Hilfe der Metadaten des zurückgesetzten Catalog-Space werden alle Anwender-Spaces der Datenbank auf der Basis des jeweils zuletzt erstellten SESAM-Sicherungsbestandes der einzelnen Anwender-Spaces ermittelt und eingespielt. Anschließend fährt SESAM/SQL die DA-LOG-Dateien, die sich an diesen SESAM-Sicherungsbestand anschließen, bis zum angegebenen Zeitpunkt nach.

ACHTUNG!
Ein Zurücksetzen auf einen zuvor erstellten SESAM-Sicherungsbestand mit RECOVER TO kann nicht rückgängig gemacht werden.

Zurücksetzen des Catalog-Space

Den Catalog-Space setzt der Datenbankverwalter mit der Utility-Anweisung RECOVER CATALOG_SPACE TO zurück.

Grundlage für das Zurücksetzen bildet ein zuvor erstellter SESAM-Sicherungsbestand des Catalog-Space. Das Zurücksetzen bringt den Catalog-Space auf den Stand, der zum Zeitpunkt der Erstellung dieses SESAM-Sicherungsbestandes vorlag.

ACHTUNG!
Wird der Catalog-Space separat zurückgesetzt, kann in der Regel auf die Anwender-Spaces nicht mehr zugegriffen werden, weil ihr Änderungszeitstempel nicht mehr mit dem im Catalog vermerkten übereinstimmt.

Die Anwender-Spaces bleiben bei RECOVER CATALOG_SPACE TO unverändert und müssen einzeln repariert werden.

Welche SESAM-Sicherungsbestände von dem Catalog-Space vorliegen, ermittelt der Datenbankverwalter mit Hilfe des Utility-Monitors aus der CAT-REC-Datei.
Den gewünschten SESAM-Sicherungsbestand kann der Datenbankverwalter über die Versionsnummer der Sicherung, den Zeitstempel oder den Dateinamen des SESAM-Sicherungsbestandes auswählen.

ACHTUNG!
Ein Zurücksetzen auf einen zuvor erstellten SESAM-Sicherungsbestand mit RECOVER TO kann nicht rückgängig gemacht werden.
Alle auf den angesprochenen Sicherungsbestand folgenden Datensätze werden in der CAT-REC-Datei gelöscht. Ein neuerer Stand kann nicht mehr erreicht werden. Sie können jedoch vor dem Zurücksetzen die CAT-REC-Datei und die CAT-LOG-Dateien außerhalb von SESAM/SQL sichern.

Konsistenz der Datenbank beim Zurücksetzen

Beim Zurücksetzen einzelner Anwender-Spaces, einer Space-Liste, eines Space-Sets oder des Catalog-Spaces ist die Konsistenz der gesamten Datenbank nicht in jedem Fall garantiert.

Konsistenz beim Zurücksetzen bedeutet dabei im Einzelnen:

- Die Übereinstimmung der Metadaten im Catalog-Space (Tabellendefinitionen, Indexdefinitionen) mit der Struktur der Anwenderdaten des zurückgesetzten Anwender-Space.
- Die Übereinstimmung der Anwenderdaten in einer Tabelle mit den zugehörigen Indizes.
- Die Einhaltung der für die Tabellen definierten Integritätsbedingungen.

Wird der Catalog-Space separat zurückgesetzt, bleiben die Anwender-Spaces unverändert und müssen einzeln repariert werden. In der Regel kann auf die Anwender-Spaces nicht mehr zugegriffen werden, weil ihr Änderungszeitstempel nicht mehr mit dem im Catalog vermerkten übereinstimmt.

Werden einzelne Anwender-Spaces, eine Space-Liste oder ein Space-Set zurückgesetzt, erzwingt SESAM/SQL die Konsistenz zwischen den Metadaten im Catalog-Space und den Anwenderdaten unmittelbar im Anschluss an das Zurücksetzen über die nachfolgend beschriebenen Anpassungsmechanismen.

ACHTUNG!
Die Daten einer Tabelle können bei der Anpassung an die Metadaten im Catalog verloren gehen.
Möchte der Datenbankverwalter sichergehen, dass beim Zurücksetzen kein Datenverlust auf Grund dieser Anpassungsmaßnahmen auftritt, sollte er die Datenbank stets als Ganzes sichern und zurücksetzen.

Konsistenz zwischen Catalog-Space und Anwender-Spaces

Die einen Anwender-Space beschreibenden Metadaten (Tabellendefinitionen, Indexdefinitionen) im Catalog-Space müssen konsistent sein mit der Struktur der Anwenderdaten auf diesem Anwender-Space. Diese Konsistenz ist jedoch in der Regel nicht mehr gegeben, wenn der Anwender-Space auf einen SESAM-Sicherungsbestand zurückgesetzt wird und seit Erstellen dieses SESAM-Sicherungsbestandes SQL-Anweisungen zur Verwaltung der Speicherstruktur sowie zur Schemadefinition und -verwaltung mit Auswirkungen auf diesen Anwender-Space ausgeführt worden sind.
So können z.B. seit dem Erstellen des SESAM-Sicherungsbestandes eine Basistabelle oder ein Index in einem Anwender-Space erzeugt worden sein, die dann nach dem Zurücksetzen im Anwender-Space nicht mehr vorhanden sind. Die zugehörigen Metadaten im Catalog-Space existieren aber nach wie vor. Ebenso ist es möglich, dass zwischenzeitlich eine Basistabelle oder ein Index im Anwender-Space gelöscht worden sind und damit auch

die zugehörigen Metadaten im Catalog-Space. Nach dem Zurücksetzen des Anwender-Space sind dann Basistabelle oder Index im Anwender-Space vorhanden, ohne dass es die zugehörigen Metadaten im Catalog-Space gibt.

Zur Überwachung der Konsistenz zwischen Catalog-Space und Anwender-Spaces führt SESAM/SQL in jedem Anwender-Space eine systeminterne Tabelle, die eine Beschreibung dieses Anwender-Space enthält, entsprechend den zugehörigen Metadaten im Catalog-Space. Anhand dieser Tabelle stellt SESAM/SQL Abweichungen eines Anwender-Space von der zugehörigen Space-Beschreibung im Catalog-Space fest und passt dann den Anwender-Space wie folgt an die zugehörige Beschreibung im Catalog-Space an:

- Tabellen und Indizes, die auf dem Anwender-Space nicht vorhanden sind, zu denen aber Metadaten im Catalog-Space existieren, richtet SESAM/SQL auf dem Anwender-Space neu ein. Die Tabellen enthalten keine Daten, die Indizes sind korrekt aufgebaut und ebenfalls leer.
- Tabellen und Indizes, die zwar auf einem Anwender-Space vorhanden sind, zu denen aber keine entsprechenden Metadaten im Catalog-Space mehr existieren, löscht SESAM/SQL auf dem Anwender-Space.
- Tabellen, deren Definition und damit auch deren Beschreibung im Catalog-Space geändert wurde, löscht SESAM/SQL ebenfalls und richtet sie gemäß der geänderten Definition auf dem Anwender-Space neu ein. Diese neu erzeugten Tabellen sind leer.

Sind bei der Wiederherstellung der Konsistenz zwischen Catalog-Space und Anwender-Spaces durch SESAM/SQL Daten verloren gegangen, so kann der Datenbankverwalter die neu erzeugten, zunächst leeren Tabellen wie folgt mit den ursprünglichen Daten füllen:

- Zunächst entlädt der Datenbankverwalter die Daten mit der Utility-Anweisung UNLOAD OFFLINE ... FROM COPY_FILE aus einem SESAM-Sicherungsbestand (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“).
- Anschließend lädt er diese Daten mit der Utility-Anweisung LOAD in die neu erzeugten Tabellen.

Konsistenz zwischen Basistabellen und zugehörigen Indizes

SESAM/SQL gewährleistet Konsistenz zwischen einer Basistabelle und den für sie definierten Indizes beim Zurücksetzen, indem es die Indizes neu aufbaut, falls die Basistabelle oder die partitionierte Tabelle und ein zugehöriger Index auf unterschiedlichen Anwender-Spaces liegen und der Datenbankverwalter nicht alle betroffenen Anwender-Spaces zurückgesetzt hat.
Wenn die Tabelle und die zugehörigen Indizes innerhalb derselben Space-Liste oder desselben Space-Set liegen und der Datenbankverwalter die Space-Liste oder den Space-Set zurückgesetzt hat, dann ist Konsistenz in jedem Fall gewährleistet. Ein Neuaufbau der Indizes durch SESAM/SQL findet in diesem Fall nicht statt.

Konsistenz zwischen Basistabellen und Integritätsbedingungen

Beim Zurücksetzen eines Anwender-Space prüft SESAM/SQL, ob dieser Anwender-Space Basistabellen enthält, für die Integritätsbedingungen definiert sind. In diesem Fall prüft SESAM/SQL, ob die aktuellen, im Catalog-Space definierten Integritätsbedingungen für die Basistabellen des zurückgesetzten Anwender-Space noch erfüllt sind.

Sofern eine Integritätsbedingung verletzt ist, versetzt SESAM/SQL den Anwender-Space in den Zustand „check pending“ (siehe Abschnitt „Space-Zustände nach der Ausführung von Utility-Anweisungen“ auf Seite 166).

Es ist Aufgabe des Datenbankverwalters, die Verletzung der Integritätsbedingungen zu beheben und damit den Anwender-Space aus dem Zustand „check pending“ zu befreien. Er verwendet hierzu die Utility-Anweisung CHECK CONSTRAINTS (siehe Seite 363) sowie SQL-Anweisungen zum Abfragen und Ändern von Daten, die er mit dem Pragma CHECK OFF absetzt (siehe Seite 365).

Wiederherstellen von Datenbanken in unterschiedlichen Situationen

Durch geeignete Recovery-Maßnahmen kann eine defekte SESAM/SQL-Datenbank nach vielen unterschiedlichen Fehlersituationen (siehe Seite 209) wiederhergestellt werden. In der Regel führen die verschiedenen RECOVER-Anweisungen problemlos zur Wiederherstellung der Datenbank. Daneben können besondere Problem-Situationen auftreten, in denen die Datenbank jedoch durch geeignete Maßnahmen wiederhergestellt werden kann. In manchen Situationen kommt es nach diesen Maßnahmen zu Datenverlust. In jedem Fall gewährleistet SESAM/SQL jedoch die Konsistenz zwischen Catalog-Space und Anwender-Spaces.

Wiederherstellen von Anwender-Spaces in unterschiedlichen Situationen

Tabelle 48 beschreibt neben dem Standardfall für den RECOVER SPACE unterschiedliche Problem-Situationen und entsprechende Maßnahmen, mit denen ein Anwender-Space repariert oder zurückgesetzt werden kann.

Situation	**Maßnahmen**	**Ergebnis/ Bemerkung**
Logische Datensicherung eingeschaltet; SESAM-Sicherungsbestand und zugehörige DA-LOGs verfügbar	Ordnungsgemäße Reparatur des Space: RECOVER SPACE	Space ist repariert
DA-LOG-Datei nicht zugreifbar, RECOVER SPACE abgebrochen	Falls DA-LOG-Datei verfügbar gemacht werden kann: Fortsetzen mit RECOVER RESTART	Space ist repariert
	Falls DA-LOG-Datei nicht verfügbar gemacht werden kann: Reparatur mit RECOVER ADJUST beenden	Datenverlust durch nicht verfügbare DA-LOG und folgende DA-LOGs
SESAM-Sicherungsbestand nicht mehr verfügbar	Falls anderer SESAM-Sicherungsbestand und zugehörige DA-LOGs vorhanden: mit RECOVER USING auf diesen SESAM-Sicherungsbestand aufsetzen	Space ist repariert
	Falls Eintrag für SESAM-Sicherungsbestand mit MODIFY bereits gelöscht: mit RECOVER TO COPY_FILE auf diesen SESAM-Sicherungsbestand zurücksetzen. Bandsicherungen müssen zuvor mit HSMS oder ARCHIVE eingespielt und umbenannt worden sein.	Datenverlust, da DA-LOGs nicht nachgefahren werden können

Tabelle 48: Mögliche Abläufe bei RECOVER SPACE (Teil 1 von 2)

Situation	Maßnahmen	Ergebnis/ Bemerkung
Space für RECOVER nicht zugreifbar	Datenbank mit Administrationsanweisungen in den Status FREE, dann wieder in den Status ACTIVE versetzen oder alternativ DBH herunterfahren und wieder neu starten. Den defekten Space für Diagnosezwecke sicherstellen: 1. entweder defekten Space mit BS2000-Mitteln umbenennen 2. oder defekten Space mit BS2000-COPY kopieren und anschließend den defekten Originalspace löschen RECOVER wiederholen	Space ist repariert
Space defekt, aber kein SESAM-Sicherungsbestand vorhanden	Wiederherstellen des leeren Space mit RECOVER SPACE TO COPY_FILE '*DUMMY'. Space ist wieder zugreifbar. Mit dem Space weiterarbeiten, Laden von Daten.	Zunächst vollständiger Verlust der Daten des Anwender-Space. Mit dem Space kann weiter gearbeitet werden.
	Space mit DROP SPACE FORCED entfernen.	Vollständiger Verlust der Daten des Anwender-Space. Konsistenz zwischen Anwender-Spaces und Metadaten ist wieder hergestellt.

Tabelle 48: Mögliche Abläufe bei RECOVER SPACE (Teil 2 von 2)

Wiederherstellen der Datenbank oder des Catalog-Space in unterschiedlichen Situationen

Tabelle 49 beschreibt neben dem Standardfall für den RECOVER CATALOG bzw. RECOVER CATALOG_SPACE unterschiedliche Problem-Situationen und entsprechende Maßnahmen, mit denen die Datenbank bzw. der Catalog-Space repariert oder zurückgesetzt werden kann.

Situation	**Maßnahmen**	**Ergebnis/ Bemerkung**
Logische Datensicherung eingeschaltet; SESAM-Sicherungsbestand und zugehörige CAT-LOG und DA-LOGs verfügbar	RECOVER CATALOG_SPACE bzw. RECOVER CATALOG ausführen Klausel SCOPE PENDING: Nur defekte Spaces werden repariert.	Catalog-Space bzw. Datenbank sind repariert
Logische Datensicherung ausgeschaltet für reine Index-Spaces aber eingeschaltet für alle übrigen Spaces; SESAM-Sicherungsbestand und zugehörige CAT-LOG und DA-LOGs verfügbar für alle Spaces in der logischen Datensicherung	RECOVER CATALOG ... GENERATE INDEX ON NO LOG INDEXSPACE ausführen.	Datenbank ist repariert, Indizes sind neu aufgebaut
Probleme bei der Reparatur von Spaces beim RECOVER CATALOG	Zuerst RECOVER CATALOG_SPACE, dann Wiederherstellen der einzelnen Spaces mit RECOVER SPACE	siehe Tabelle 48
CAT-LOG-Datei nicht mehr verfügbar	Falls CAT-LOG-Datei verfügbar gemacht werden kann: Wiederholen des RECOVER CATALOG_SPACE bzw. RECOVER CATALOG	Catalog-Space bzw. Datenbank sind repariert
	Falls CAT-LOG-Datei nicht verfügbar gemacht werden kann: 1. Mit RECOVER ... TO auf den jüngsten SESAM-Sicherungsbestand zurücksetzen. 2. Prüfen, ob evtl. Maßnahmen zur Erzeugung einer neuen CAT-REC- Datei sinnvoll sind. Dazu muss der Software-Kundendienst eingeschaltet werden. Ggf. RECOVER CATALOG_SPACE bzw. RECOVER CATALOG ausführen.	1. Datenverlust, da keine CAT-LOGs nachgefahren werden. 2. Datenverlust, da nur die CAT-LOGs vor der fehlenden CAT-LOG-Datei nachgefahren werden können.

Tabelle 49: Mögliche Abläufe bei RECOVER CATALOG und RECOVER CATALOG_SPACE (Teil 1 von 2)

Situation	Maßnahmen	Ergebnis/ Bemerkung
SESAM-Sicherungsbestand nicht mehr verfügbar	Falls anderer SESAM-Sicherungsbestand sowie zugehörige CAT-LOG- und DA-LOG-Dateien vorhanden: mit RECOVER USING auf diesen SESAM-Sicherungsbestand aufsetzen.	Catalog-Space bzw. Datenbank sind repariert
	Falls Eintrag für geeigneten SESAM-Sicherungsbestand bereits aus der CAT-REC-Datei gelöscht: Prüfen, ob evtl. Maßnahmen zur Erzeugung neuer CAT-REC-Datei sinnvoll. Dazu muss der Software-Kundendienst kontaktiert werden.	Vollständiger Datenverlust, da Reparatur nur mit Dateien möglich, die in CAT-REC verzeichnet sind.
Catalog-Space für RECOVER CATALOG[_SPACE] nicht zugreifbar	1. Datenbank mit Administrationsanweisungen in den Status FREE, dann wieder in den Status ACTIVE versetzen 2. oder DBH herunterfahren und wieder neu starten Den defekten Catalog-Space für Diagnosezwecke sicherstellen und löschen. Catalog wieder in das SQL-Datenbankverzeichnis eintragen bzw. DBH wieder starten und RECOVER CATALOG[_SPACE] wiederholen.	Catalog-Space ist repariert

Tabelle 49: Mögliche Abläufe bei RECOVER CATALOG und RECOVER CATALOG_SPACE (Teil 2 von 2)

Wiederherstellen der Datenbank bei defekter CAT-REC-Datei

Die CAT-REC-Datei sollte möglichst parallel geführt werden (siehe „Sichern der CAT-REC-Datei der Originaldatenbank“ auf Seite 217). Falls die CAT-REC-Datei dennoch einmal defekt ist, muss der Software-Kundendienst kontaktiert werden.

7.5.8 Indizes wiederherstellen

Defekte Indizes kann der Datenbankverwalter mit der Utility-Anweisung RECOVER INDEX wiederherstellen bzw. neu aufbauen (siehe Seite 377).

Das Wiederherstellen bzw. der Neuaufbau erfolgt wahlweise

- für einen bestimmten Index, der über den Indexnamen identifiziert wird,
- für alle Indizes, die zu einer bestimmten Tabelle gehören,
- für alle Indizes, die zu Tabellen eines bestimmten Anwender-Space gehören.

7.6 Replikat einer Datenbank

Ein Replikat ist eine Kopie einer Datenbank, die für die Wiedergewinnung oder als Schattendatenbank für die Recovery verwendet werden kann.

Replikate sind in demselben DBH wie das Original unter einem anderen Namen oder in einem anderen DBH unter beliebigem Namen ablauffähig. Für lesende Zugriffe kann ein Replikat in das SQL-Datenbankverzeichnis mehrerer DBHs eingetragen sein.

Der Datenbankverwalter kann Replikate einer Originaldatenbank aus SESAM-Sicherungsbeständen (Catalog-Sicherungen auf Platte oder Magnetbandkassette) oder einer Fremdkopie (siehe Abschnitt „Replikate aus Fremdkopien erstellen“ auf Seite 391) erstellen und in beliebigen Abständen aktualisieren. Bei Bedarf können Spaces aus einem Replikat entfernt oder neu hinzugefügt werden.

Ein Replikat, das alle Spaces der Original-Datenbank (Catalog-Space und alle Anwender-Spaces) enthält, wird als vollständiges Replikat bezeichnet. Ein Teilreplikat beinhaltet den Catalog-Space und wenigstens einen der Anwender-Spaces.
Ein vollständiges Replikat kann zu einem Teilreplikat werden. Umgekehrt kann ein Teilreplikat zu einem vollständigen Replikat ergänzt werden.

Replikate und Teilreplikate können zum Lesen verwendet werden. Wenn ein Replikat eine partitionierte Tabelle enthält, dann müssen alle Partitionen der Tabelle im Replikat enthalten sein. Ein Teilreplikat muss immer in sich abgeschlossen sein, d.h. das Teilreplikat muss alle Indizes zu allen Tabellen beinhalten, die zum Teilreplikat gehören.

Ein vollständiges Replikat entsteht aus einer kompletten Catalog-Sicherung durch die Utility-Anweisung CREATE REPLICATION.

Ein Teilreplikat entsteht durch

- die Utility-Anweisung CREATE REPLICATION ... FOR SPACE ... aus einer kompletten Catalog-Sicherung oder einer Catalog-Sicherung ohne NO LOG Index-Spaces
- die Utility-Anweisung CREATE REPLICATION ... aus einer Catalog-Sicherung ohne NO LOG Index-Spaces
- die Utility-Anweisung REFRESH REPLICATION ... FOR SPACE ... aus einem vollständigen Replikat oder einem Teilreplikat z.B. dann, wenn für einen Replikat-Space im Original das Logging unterbrochen war und dieser Space im Replikat nicht mehr aktualisiert werden kann
- die SQL-Anweisung CREATE SPACE, mit der im Original ein neuer Space erzeugt wird

Der Datenbankverwalter kann ein Replikat mit der Utility-Anweisung REFRESH SPACE um Anwender-Spaces des Originals auf Basis einer Sicherung erweitern. Diese Spaces können neu im Replikat sein oder bereits früher zum Replikat gehört haben („ehemalige Replikat-Spaces“). Das Replikat muss soweit aktualisiert sein, dass die verwendete Sicherung im Replikat bekannt ist.

Informationen über die aktuellen und ehemaligen Replikat-Spaces erhalten Sie über den Utility-Monitor, Maske COP.4.6 (siehe Handbuch „Utility-Monitor“).

Um ein Replikat mit REFRESH REPLICATION aktualisieren zu können, müssen alle Spaces des Replikats im Logging sein und das Logging darf nicht unterbrochen worden sein (z.B. durch LOAD OFFLINE). Dies gilt auch für Spaces, die mit REFRESH SPACE einem Replikat hinzugefügt werden sollen.

Im Rahmen der Media-Recovery kann der Datenbankverwalter als Alternative zu SESAM-Sicherungen auch ein Replikat zum Reparieren oder Zurücksetzen einer Originaldatenbank verwenden:

- Er kann Anwender-Spaces, den Catalog-Space oder die gesamte Datenbank mit Hilfe des Replikats reparieren (siehe Abschnitt „Originaldatenbank mit einem Replikat reparieren“ auf Seite 256). Das geht unter bestimmten Voraussetzungen schneller als ein normaler RECOVER.
- Er kann Anwender-Spaces, den Catalog-Space oder die gesamte Datenbank auf den Stand des Replikats zurücksetzen (siehe Abschnitt „Zurücksetzen der Originaldatenbank mit Hilfe eines Replikats“ auf Seite 260).
- Er kann aus einem Replikat eine neue Originaldatenbank erzeugen (siehe Abschnitt „Eigenständige Originaldatenbank aus dem Replikat erzeugen“ auf Seite 263).

7.6.1 Replikat erzeugen

- Replikat aus SESAM/SQL-Sicherungsbestand erstellen

  Mit der Utility-Anweisung CREATE REPLICATON erstellt der Datenbankverwalter aus einem SESAM-Sicherungsbestand, der mit COPY CATALOG generiert wurde, ein Replikat. Das Replikat entspricht der Originaldatenbank zum Zeitpunkt der Kopie.
  Die CAT-REC-Datei, die beim CREATE REPLICATION angegeben wird, identifiziert eine Sicherungskopie. Sie muss die gewünschte Datenbank-Sicherung als Letzte enthalten. Die CAT-REC-Kopie, die beim COPY CATALOG entstanden ist, genügt dieser Anforderung.
  Die CAT-REC-Datei des Replikats, die beim CREATE REPLICATION entsteht, beschreibt den Stand des Originals zum Zeitpunkt des COPY CATALOG.

  Die für CREATE REPLICATION verwendete Sicherung muss immer eine vollständige, mit COPY CATALOG entstandene Datenbank-Sicherung auf Magnetbandkassette oder Platte sein.

- Replikat aus SESAM-fremder Kopie erstellen

  Der Datenbankverwalter kann ein Replikat oder Teilreplikat auch aus einer Fremdkopie erstellen. Erläuterungen zum Generieren einer Fremdkopie finden Sie im Abschnitt „Fremdkopien einer Datenbank einsetzen“ auf Seite 382.

Ein Teilreplikat kann erzeugt werden, indem die gewünschten Anwender-Spaces mit dem Parameter FOR SPACE spezifiziert werden. Der Catalog-Space ist immer im Replikat enthalten. Das erzeugte Replikat kann später nur dann aktualisiert werden, wenn alle Spaces des Replikats im Original mit Logging betrieben werden.

Nach dem Erstellen kann das Replikat sofort für die Wiedergewinnung verwendet werden. Es ist automatisch im SQL-Datenbankverzeichnis des DBH eingetragen.

Beim Starten des DBH kann es wie jede andere Datenbank ins SQL-Datenbankverzeichnis mit dem Parameter ACCESS=REPLICATION bei ADD-SQL-DATABASE-CATALOG-LIST eingetragen oder bei laufendem DBH mit dem gleichen Parameter der Administrationsanweisung ADD-SQL-DB-CATALOG-ENTRY eingehängt werden (siehe Handbuch „Datenbankbetrieb“). Wenn Originaldatenbank und Replikat unter verschiedenen DBHs arbeiten, können sie den gleichen logischen Namen verwenden.

Ein Replikat kann im SQL-Datenbankverzeichnis verschiedener DBHs für die Wiedergewinnung eingetragen sein.

Für SESDCN gilt:
Da die Beziehung zwischen Originalname und Name des Replikats erst im DBH hergestellt wird, können bei gleichzeitiger Verwendung von Original und Replikat an 2 DBHs nicht beide an der Verteilung teilnehmen, falls beide mit dem gleichen logischen Datenbanknamen angesprochen werden sollen.

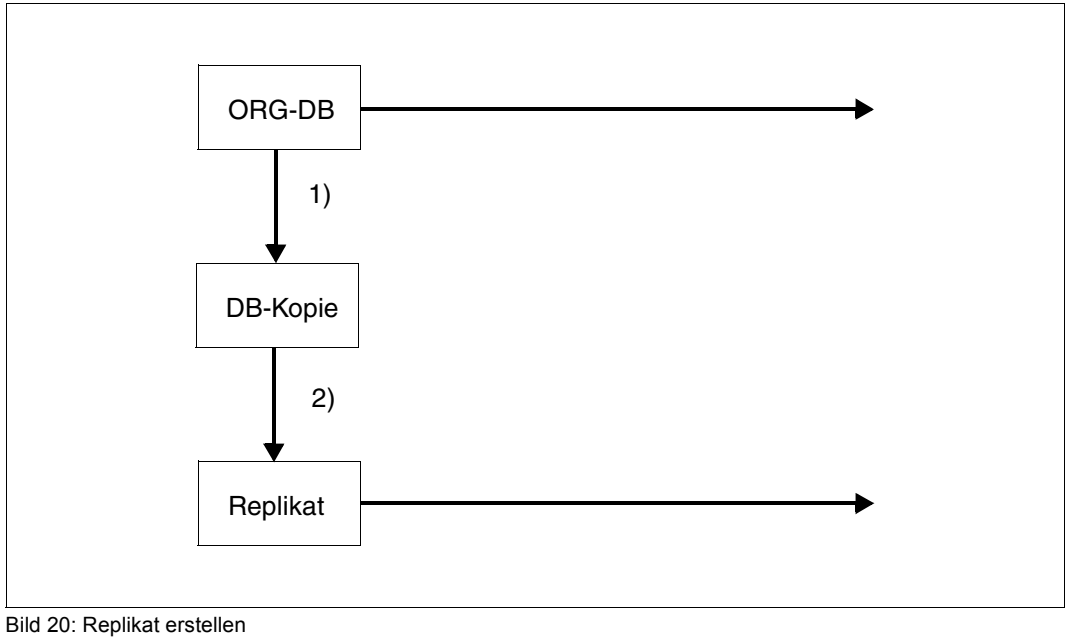


Bild 20: Replikat erstellen

1) Erzeugen eines SESAM-Sicherungsbestandes der Originaldatenbank
2) Erstellen des Replikats aus der Kopie

7.6.2 Wiedergewinnung mit einem Replikat

Ein wesentliches Einsatzgebiet für Replikate bzw. Teilreplikate ist die Entlastung eines DBH im OLTP-Betrieb von rein lesenden Anwendungen, die nicht unbedingt den aktuellen Stand der Daten benötigen (z.B. erstellen von Auswertungen oder Statistiken).

Wiedergewinnung mit einem Teilreplikat

Der Datenbankverwalter kann festlegen, dass nur ein Teilreplikat verwendet werden soll. Dieses Teilreplikat enthält nur die Teile der Datenbank, die in der aktuellen Anwendung gebraucht werden. Es benötigt weniger Platz und ist schneller erstellt und aktualisiert.

Ein Teilreplikat muss immer in sich abgeschlossen sein, d.h. das Teilreplikat muss alle Indizes zu allen Tabellen beinhalten, die zum Teilreplikat gehören.

SESAM/SQL reagiert mit dem SQL-Statuscode „Index bzw. Space nicht zugreifbar“, wenn Indizes oder Tabellen angesprochen werden, die nicht Bestandteil des Teilreplikats sind.

7.6.3 Replikat aktualisieren und erweitern

Mit der Utility-Anweisung REFRESH REPLICATION aktualisiert der Datenbankverwalter ein Replikat oder ein Teilreplikat:

- Mit REFRESH REPLICATION
  wird das gesamte Replikat bzw. Teilreplikat durch Nachfahren der Logging-Dateien aktualisiert.
- Mit REFRESH REPLICATION FOR SPACE
  werden nur die angegebenen Spaces und der Catalog-Space des Replikats durch Nachfahren der Logging-Dateien aktualisiert. Das Replikat enthält dann nur noch diese Spaces und wird zu einem Teilreplikat.

Ein Replikat sollte in folgenden Fällen aktualisiert werden:

- Wenn eine Wiedergewinnungsanwendung auf möglichst aktuelle Werte zugreifen soll.
- Wenn es als Schattendatenbank für ein besonders schnelles Reparieren verwendet werden soll.

Der Datenbankverwalter kann ein Replikat mit der Utility-Anweisung REFRESH SPACE um Anwender-Spaces des Originals auf Basis einer Sicherung erweitern. Diese Spaces können neu im Replikat sein oder bereits früher zum Replikat gehört haben („ehemalige Replikat-Spaces“). Das Replikat muss soweit aktualisiert sein, dass die verwendete Sicherung in den Metadaten des Replikats bekannt ist.
Informationen über die aktuellen und ehemaligen Replikat-Spaces erhalten Sie über den Utility-Monitor, Maske COP.4.6 (siehe Handbuch „Utility-Monitor“).

Während eines REFRESH REPLICATION oder REFRESH SPACE ist kein Zugriff aus lesenden Anwendungen heraus auf das Replikat möglich, da für die Aktualisierung ein exklusiver Zugriff notwendig ist. Das Replikat darf nur in das SQL-Datenbankverzeichnis des DBH eingetragen sein, an dem die Utility-Anweisung durchgeführt wird.

7.6.3.1 Replikat aktualisieren (REFRESH REPLICATION)

REFRESH REPLICATION aktualisiert das gesamte Replikat mit allen Anwender-Spaces, die aktuell zum Replikat gehören.

REFRESH REPLICATION FOR SPACE aktualisiert den Catalog-Space des Replikats und alle angegebenen Anwender-Spaces. Falls das Replikat mehr Anwender-Spaces enthalten hat als angegeben, werden diese logisch aus dem Replikat entfernt („ehemalige Replikat-Spaces“). Das Replikat wird zu einem Teilreplikat des ursprünglichen Replikats. Ehemalige Replikat-Spaces werden nicht physikalisch gelöscht.

Der Datenbankverwalter muss alle Logging-Dateien einer bestimmten Zeitspanne und die CAT-REC-Kopie der Originaldatenbank bereitstellen. Die Jobvariable SESAM.*replikat*.NEXT-REPL-LOG enthält Nummer und Unternummer der jeweils ersten bereitzustellenden Dateien. Die Jobvariable wird bei CREATE REPLICATION angelegt und bei REFRESH REPLICATION jeweils aktualisiert.

Alle Spaces eines Replikats müssen im Original mit Logging betrieben werden. Spaces, bei denen im Original das Logging unterbrochen wird, können nicht mehr aktualisiert werden. In diesem Fall kann das Replikat mit REFRESH REPLICATION FOR SPACE verkleinert werden, wenn der betreffende Space nicht in der Space-Liste bei FOR SPACE angegeben wird.
Das Logging wird unterbrochen durch die Anweisungen ALTER SPACE NO LOG, LOAD OFFLINE, IMPORT TABLE, RECOVER INDEX, RECOVER TO und MIGRATE, siehe auch „Unterbrechen des Logging durch SESAM/SQL bei einigen Utility-Anweisungen“ auf Seite 229.

Ein Replikat kann auch dann mit REFRESH REPLICATION FOR SPACE verkleinert werden, wenn Spaces des Replikats ungültig sind oder nicht mehr im Replikat benötigt werden. Spaces werden ungültig durch DROP SPACE im Original, RECOVER SPACE USING REPLICATION RENAME, RECOVER SPACE TO REPLICATION oder RECOVER SPACE USING REPLICATION TO *marke*.

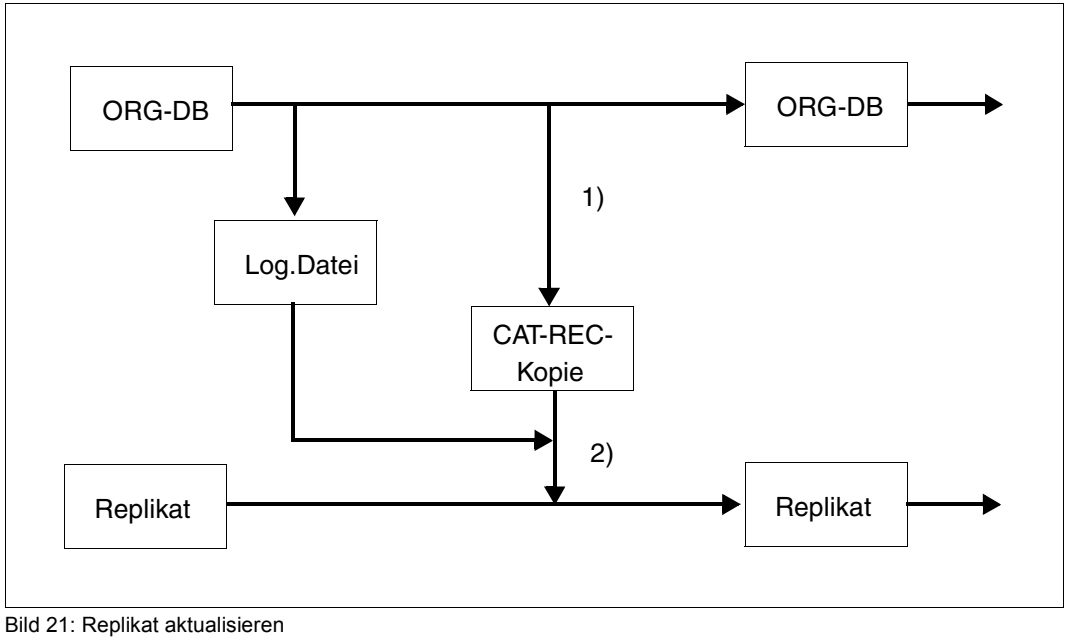


Bild 21: Replikat aktualisieren

1) Wechsel der Logging-Dateien und Erzeugen einer CAT-REC-Kopie
2) Aktualisieren des Replikats mit Hilfe von Logging-Dateien und der CAT-REC-Kopie

Vorgehensweise

Zu einem definierten Zeitpunkt soll das Replikat auf den Stand der Originaldatenbank gebracht werden. Dazu ist es notwendig, mit CHANGE-CATLOG oder
COPY CATALOG_SPACE die CAT-LOG- und DA-LOG-Dateien auf dem Original zu wechseln. Dabei wird eine CAT-REC-Kopie erzeugt.

Es ist sinnvoll, den CHANGE-CATLOG im transaktionslosen Zustand abzusetzen, und zwar aus folgendem Grund:

Wenn noch Transaktionen aktiv sind, werden diese am Ende der Aktualisierung im Replikat zurückgesetzt.

Sind Transaktionen zurückgesetzt worden, müssen bei einem späteren REFRESH REPLICATION wieder alle Logging-Dateien ab dem Transaktionsbeginn der zurückgesetzten Transaktionen abgearbeitet werden. Nur so kann ein definierter Zustand im Replikat erreicht werden.
Ein REFRESH REPLICATION sollte daher vermieden werden, wenn langlaufende Transaktionen offen sind. Im ungünstigsten Fall werden diese u.U. mehrmals nachgefahren, zurückgesetzt und wieder nachgefahren.

Beim REFRESH REPLICATION wird auf der CAT-REC-Datei des Replikats aufgesetzt. Aus der Differenz zur CAT-REC-Kopie wird deutlich, welche CAT-LOG-Dateien in den Catalog-Space eingearbeitet werden müssen. Nach dem Einarbeiten der CAT-LOG-Dateien kann mit den aktualisierten Catalog-Tabellen RECOVERY_UNITS und DA-LOGS des bearbeiteten Catalog-Space bestimmt werden, welche DA-LOG-Dateien in den Anwender-Spaces nachgefahren werden müssen. Dann werden die DA-LOG-Dateien nachgefahren.

Damit hat das Replikat den gleichen Stand wie das Original zum Zeitpunkt der verwendeten CAT-REC-Kopie.

In einer Jobvariable werden die Nummern der ältesten Logging-Dateien eingetragen, die für den nächsten REFRESH REPLICATION nötig sind.

Der Catalog-Space und die Anwender-Spaces werden wie bei RECOVER in der Service-Task in einem Nachfahr-DBH aktualisiert.
Das Replikat wird gesperrt, wenn beim Nachfahren ein Fehler auftritt, z.B. weil eine Logging-Datei fehlt, ein Speicherengpass auftritt oder die Service-Task beendet wird. Wenn die Fehlerursache beseitigt worden ist, kann das Replikat mit REFRESH REPLICATION fertiggestellt werden.
Wenn REFRESH REPLICATION abbricht, weil der DBH beendet wird oder der DBH einen internen Wiederanlauf durchführt, ist eine Wiederholung des REFRESH REPLICATION nicht möglich. In diesem Fall wird das Replikat als defekt markiert und muss neu erzeugt werden.

CREATE SPACE im Original macht ein existierendes Replikat automatisch zum Teilreplikat, da der neue Space nicht im Replikat enthalten ist.

7.6.3.2 Replikat erweitern (REFRESH SPACE)

Der Datenbankverwalter kann ein Replikat mit der Utility-Anweisung REFRESH SPACE um die angegebenen Anwender-Spaces erweitern. Diese Spaces können neu im Replikat sein. Sie können auch bereits früher zum Replikat gehört haben („ehemalige Replikat-Spaces“), wurden dann aber aus dem Replikat entfernt, weil das Logging unterbrochen war oder weil sie ungültig waren.

REFRESH SPACE fügt die angegebenen Spaces aus einem SESAM-Sicherungsbestand (Platte oder Magnetbandkassette) oder einer Fremdkopie dem Replikat hinzu und bringt sie durch Nachfahren der Logging-Dateien auf den aktuellen Stand des Replikats.

REFRESH SPACE prüft nicht, ob angegebene Spaces bereits im Replikat enthalten sind. Bereits vorhandene Spaces werden überschrieben. Wird ein Replikat-Space neu angelegt, so wird das Medium des Catalog-Space des Replikats verwendet.

Das Replikat muss für REFRESH SPACE soweit aktualisiert sein, dass die verwendeten Sicherungen im Replikat bekannt sind. Es können auch Sicherungen verwendet werden, die zu einem Zeitpunkt entstanden, der älter ist als die Sicherung aus der das Replikat erzeugt wurde.
Werden bei REFRESH SPACE mehrere Spaces angegeben, dann müsses alle verwendeten Sicherungen mit dem gleichen Zeitstempel versehen sein.
Spaces werden von REFRESH SPACE bis auf den aktuellen Stand des Replikats nachgefahren. Zwischen der verwendeten Sicherung und dem Stand des Replikats darf auf den betroffenen Spaces im Original das Logging nicht unterbrochen worden sein.

Der Datenbankverwalter muss alle DA-LOG-Dateien bereitstellen, die seit dem Zeitpunkt der Sicherung bis zum aktuellen Stand des Replikats entstanden sind. Welche Dateien benötigt werden, kann über die RECOVERY_UNITS im INFORMATION_SCHEMA festgestellt werden.

7.6.4 Originaldatenbank mit einem Replikat reparieren

7.6.4.1 RECOVER CATALOG USING REPLICATION

Mit der Utility-Anweisung RECOVER CATALOG USING REPLICATION kann der Datenbankverwalter eine komplette Originaldatenbank mit einem vollständigen Replikat reparieren. Mit Teilreplikaten können die entsprechenden Teile des Originals repariert werden.

Der Stand, der mit der Reparatur erreicht wird, wird durch die angegebene CAT-REC-Datei definiert. Die angegebene CAT-REC-Datei darf nicht älter sein als die des Replikats.
Wird eine ältere CAT-REC-Datei angegeben als die aktuelle CAT-REC-Datei des Originals, handelt es sich eigentlich um ein Zurücksetzen der Datenbank (siehe Abschnitt „Zurücksetzen mit RECOVER CATALOG USING REPLICATION“ auf Seite 261).

Die CAT-REC-Datei der Originaldatenbank wird bei RECOVER USING REPLICATION überschrieben. Der Datenbankverwalter muss den Stand des Originals vor dem RECOVER sichern, wenn er eventuell noch einmal auf ihn zurückgreifen will.

Die Reparatur mit einem Replikat ist nur möglich, wenn Catalog-Space und Anwender-Spaces der Originaldatenbank immer mit Logging bearbeitet wurden.
Ein Replikat, das regelmäßig aktualisiert wird, erlaubt ein sehr viel schnelleres Reparieren der Originaldatenbank. Bei einer Reparatur der Originaldatenbank mit Hilfe des Replikats müssen nur die Logging-Dateien nachgefahren werden, die sich aus dem Unterschied zwischen Original und Replikat zum Zeitpunkt eines Fehlers ergeben.
Je häufiger ein Replikat aktualisiert wird, desto aktueller ist es und desto schneller kann dann eine Reparatur durchgeführt werden.

Existiert nur ein Teilreplikat, können auch nur der Catalog-Space und die Anwender-Spaces repariert werden, die im Teilreplikat enthalten sind. Die anderen Spaces müssen dann eventuell mit RECOVER SPACE repariert werden (siehe „Wiederherstellen von Anwender-Spaces in unterschiedlichen Situationen“ auf Seite 243).

Reparatur mit einem Replikat ist nur möglich, wenn auf dem Original die Bedingungen eingehalten worden sind, die auch einen REFRESH REPLICATION erlauben. Das bedeutet, die logische Datensicherung darf nicht unterbrochen und es darf kein Space des Replikats im Original mit DROP SPACE gelöscht worden sein.

Bei der Funktion COPY bleibt das Replikat erhalten und kann weiter prozessiert werden. Bei der Funktion RENAME wird das Replikat zur Originaldatenbank und ist anschließend nicht mehr vorhanden. Replikat und Original müssen bei RENAME auf derselben BS2000-Kennung liegen und die Dateien des Originals dürfen nicht existieren, da sonst das Umkatalogisieren der Dateien nicht möglich ist.

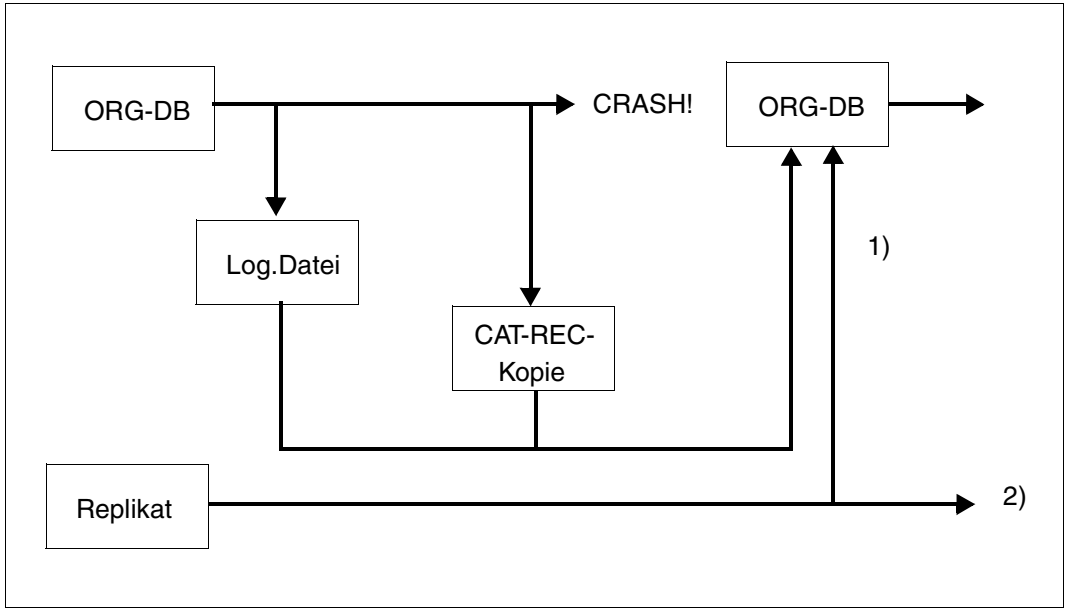


Bild 22: Originaldatenbank mit Hilfe eines Replikats reparieren

1) Beim RECOVER CATALOG USING REPLICATION kann eine Datenbank aus dem Replikat, den Logging-Dateien und der aktuellen CAT-REC-Datei repariert werden.

2) Bei der Option COPY bleibt das Replikat unverändert bestehen.

Vorgehensweise

Ist die CAT-REC-Datei der Originaldatenbank bei einem Fehler nicht physikalisch zerstört worden, kann sie für die Reparatur verwendet werden. Ist keine CAT-REC-Datei mehr vorhanden, muss die jüngste Kopie bestimmt werden.

Um den aktuellen Stand eines Originals wiederherstellen zu können, muss die CAT-REC-Datei des Originals für die Reparatur verwendet werden. Wird eine ältere CAT-REC-Kopie verwendet, so bedeutet das ein Zurücksetzen des Catalogs (siehe Beispiel auf Seite 261).

Der Datenbankverwalter muss für die Reparatur die benötigten Logging-Dateien zur Verfügung stellen. Wird das Replikat vom gleichen DBH prozessiert wie die Originaldatenbank, dann muss es vor der Reparatur in den Status FREE versetzt werden. Bei unterschiedlichen DBHs kann dies entfallen. Es darf aber kein paralleler REFRESH REPLICATION des Replikats stattfinden.

Beim Abbruch eines RECOVER USING REPLICATION, bei dem nicht die aktuelle CAT-REC-Datei angegeben ist, ist eventuell schon die CAT-REC-Datei des Originals überschrieben. Um sicherzustellen, dass nichts verloren geht, sollte diese Datei vor dem RECOVER gesichert werden. Dies wird von SESAM/SQL nicht automatisch gemacht.

Teilreplikat ohne Index-Space

Wird das Teilreplikat nicht für Wiedergewinnung verwendet, kann auf die Indizes zu den Tabellen verzichtet werden. Es ist dann z.B. möglich, nur alle Spaces mit den Tabellen als Teilreplikat zu führen. Bei einer Reparatur mit diesem Teilreplikat werden die Spaces mit den Primärdaten wiederhergestellt. Die Spaces mit den Indizes können dann über RECOVER TO *DUMMY neu generiert werden.

Index-Spaces außerhalb des Teilreplikats können so auch ohne Logging sein. Diese Spaces werden dann auf einen alten Sicherungsstand zurückgesetzt oder mit RECOVER TO *DUMMY neu generiert, wobei alle Indizes neu aufgebaut werden. Dieses Vorgehen ist allerdings aus Zeitgründen nur bei kleinen Tabellen oder wenigen Indizes zu empfehlen.

7.6.4.2 RECOVER CATALOG_SPACE USING REPLICATION

Mit der Utility-Anweisung RECOVER CATALOG_SPACE USING REPLICATION kann der Datenbankverwalter den Catalog-Space des Originals reparieren. Das Replikat kann ein vollständiges Replikat oder ein beliebiges Teilreplikat sein, da der Catalog-Space stets im Replikat enthalten ist.

Für die Logging-Dateien und die CAT-REC-Datei gelten die gleichen Aussagen wie im Abschnitt „RECOVER CATALOG USING REPLICATION“ auf Seite 256, wobei aber nur die CAT-LOG-Dateien benötigt werden.

Bei RECOVER CATALOG_SPACE USING REPLICATION mit RENAME ist anschließend der Catalog-Space des Replikats nicht mehr vorhanden. Das Replikat muss neu erzeugt werden.

7.6.4.3 RECOVER SPACE USING REPLICATION

Mit der Utility-Anweisung RECOVER SPACE USING REPLICATION kann der Datenbankverwalter einen oder mehrere Anwender-Spaces des Originals reparieren. Das Replikat kann ein vollständiges Replikat oder ein beliebiges Teilreplikat sein. Die zu reparierenden Anwender-Spaces müssen im Replikat aktuell enthalten sein.

Für die Logging-Dateien gelten die gleichen Aussagen wie im Abschnitt „RECOVER CATALOG USING REPLICATION“ auf Seite 256, wobei aber nur die DA-LOG-Dateien benötigt werden. Aussagen, die dort über die CAT-REC-Datei getroffen werden, sind hier ohne Bedeutung, weil der Catalog-Space nicht betroffen ist.

Bei RECOVER SPACE USING REPLICATION mit RENAME sind anschließend die betroffenen Anwender-Spaces im Replikat nicht mehr vorhanden. Diese Spaces können mit REFRESH SPACE dem Replikat wieder hinzugefügt werden.

7.6.5 Zurücksetzen der Originaldatenbank mit Hilfe eines Replikats

7.6.5.1 RECOVER CATALOG TO REPLICATION

Mit der Utility-Anweisung RECOVER CATALOG TO REPLICATION kann der Datenbankverwalter eine gesamte Originaldatenbank mit einem vollständigen Replikat auf den Stand des Replikats zurücksetzen.

Dabei werden auch die CAT-REC-Datei und die Metadaten angepasst. Der Datenbankverwalter muss den Stand vor dem RECOVER sichern, wenn er eventuell noch auf diesen Stand zurückgreifen will.

Nach dem Zurücksetzen eines Originals werden alte Logging-Dateien überschrieben. Der Stand vor dem Zurücksetzen kann dann mit RECOVER nicht mehr erreicht werden.

Bei der Funktion COPY bleibt das Replikat erhalten und kann weiter prozessiert werden. Bei der Funktion RENAME wird das Replikat zum Original und ist anschließend nicht mehr vorhanden. Replikat und Original müssen auf der DBH-Kennung liegen und die Dateien des Originals dürfen nicht existieren, da sonst das Umkatalogisieren der Dateien nicht möglich ist.

Bei Verwendung eines Teilreplikates werden nur die Spaces zurückgesetzt, die im Teilreplikat enthalten sind. Dies sind der Catalog-Space und die Anwender-Spaces, die im Teilreplikat aktuell sind. Alle anderen Anwender-Spaces des Originals müssen anschließend an den zurückgesetzten Stand angepasst werden.
Bei NO LOG Index-Spaces kann dies automatisch durch RECOVER CATALOG TO REPLICATION mit Angabe von GENERATE INDEX ON NO LOG INDEX SPACE geschehen. Alle anderen Spaces müssen einzeln mit RECOVER SPACE angepasst werden.

7.6.5.2 Zurücksetzen mit RECOVER CATALOG USING REPLICATION

Ein Sonderfall ist das Zurücksetzen mit RECOVER CATALOG USING REPLICATION und der Angabe einer CAT-REC-Datei, die genauso alt oder jünger ist als das Replikat, aber älter als der aktuelle Stand des Originals. Dabei wird das Original nicht genau auf den Stand zurückgesetzt, den der Catalog hatte als die CAT-REC-Kopie entstand, sondern es wird ein CAT-LOG-Eintrag mehr nachgefahren. Auf diese Weise kann man mit der letzten CAT-REC-Kopie dem aktuellen Stand näher kommen oder ihn sogar erreichen, wenn die CAT-REC-Datei des Originals defekt ist.

Beispiel

Bei der Erzeugung einer CAT-REC-Kopie wird bereits die neue, gewechselte CAT-LOG-Datei eingetragen. Ein REFRESH REPLICATION berücksichtigt diese Tatsache und wertet diesen letzten Eintrag nicht aus, da es sich normalerweise um die aktuelle und damit offene CAT-LOG-Datei handelt.

Führt man jedoch einen RECOVER USING REPLICATION mit Angabe dieser CAT-REC-Kopie aus, dann wird dieser letzte Eintrag ausgewertet, was in diesem Fall bedeutet, dass alles nachgefahren wird, was in dieser CAT-LOG-Datei und darin enthaltenen DA-LOGS protokolliert wurde. Falls es keinen CAT-LOG-Wechsel gab, wird alles nachgefahren. Bei einem CAT-LOG-Wechsel endet das Nachfahren an dem Punkt, an dem der Wechsel stattfand.

7.6.5.3 RECOVER CATALOG_SPACE TO REPLICATION

Mit der Utility-Anweisung RECOVER CATALOG_SPACE TO REPLICATION kann der Datenbankverwalter den Catalog-Space des Originals auf den Stand des Replikats zurücksetzen. Das Replikat kann ein vollständiges Replikat sein. Es kann auch ein beliebiges Teilreplikat sein, da der Catalog-Space stets im Replikat enthalten ist.

Alle Anwender-Spaces des Originals müssen anschließend einzeln mit RECOVER SPACE an den zurückgesetzten Stand angepasst werden.

Bei RECOVER CATALOG_SPACE TO REPLICATION mit RENAME ist anschließend der Catalog-Space des Replikats nicht mehr vorhanden. Das Replikat muss neu erzeugt werden.

7.6.5.4 RECOVER SPACE TO REPLICATION

Mit der Utility-Anweisung RECOVER SPACE TO REPLICATION kann der Datenbankverwalter Anwender-Spaces des Originals zurücksetzen. Das Replikat kann ein vollständiges Replikat oder ein beliebiges Teilreplikat sein. Die zurückzusetzenden Anwender-Spaces müssen im Replikat aktuell enthalten sein.

Für den Indexaufbau auf Index-Spaces, die durch das Rücksetzen eines Table-Spaces betroffen sind, gelten die gleichen Aussagen wie beim Rücksetzen von Spaces mit SESAM-Sicherungen oder Fremdkopien. Mit der Klausel NO INDEX (siehe Seite 232) kann der Indexaufbau auf abhängigen Indexspaces verhindert werden; betroffene Indizes sind anschließend defekt.

Bei RECOVER SPACE USING REPLICATION mit RENAME sind anschließend die betroffenen Anwender-Spaces im Replikat nicht mehr vorhanden.
Bei RECOVER SPACE USING REPLICATION mit COPY bleiben die betroffenen Anwender-Spaces im Replikat zwar erhalten, können aber im Replikat nicht mehr aktualisiert werden, weil das Logging durch das Rücksetzen im Original unterbrochen wird.

Um zurückgesetzte Spaces dem Replikat wieder hinzuzufügen, sind folgende Schritte notwendig:

1. Spaces im Original kopieren
2. Neue CAT-REC.COPY-Datei mit der Administrationsanweisung CHANGE-CATLOG erzeugen
3. Replikat aktualisieren mit REFRESH REPLICATION FOR SPACE ohne Angabe der zurückgesetzten Spaces
4. Zurückgesetzte Spaces mit REFRESH SPACE aus der zuvor erzeugten Sicherung wieder dem Replikat hinzufügen

7.6.5.5 RECOVER SPACE USING REPLICATION TO *marke*

Mit der Utility-Anweisung RECOVER SPACE USING REPLICATION TO *marke* kann der Datenbankverwalter einen Anwender-Space des Originals zurücksetzen und bis zur angegebenen Marke nachfahren. Eine Marke repräsentiert den Zeitpunkt einer Unterbrechung des Logging. Sie entsteht z.B. bei LOAD OFFLINE vor dem Laden der Anwenderdaten.

Es gelten die gleichen Aussagen wie bei RECOVER SPACE TO REPLICATION.

7.6.6 Eigenständige Originaldatenbank aus dem Replikat erzeugen

Sowohl mit RECOVER CATALOG USING REPLICATION als auch mit RECOVER CATALOG TO REPLICATION kann eine neue Originaldatenbank erzeugt werden. Die neue Datenbank muss im SQL-Datenbankverzeichnis eingetragen werden. RECOVER CATALOG USING/TO REPLICATION erzeugt aus dem Replikat alle Spaces der neuen Datenbank, sofern diese auch aktuell Teil des Replikats waren.

Vorgehensweise

Der Datenbankverwalter trägt mit der Administrationsanweisung ADD-SQL-DB-CATALOG-ENTRY die neue Datenbank im DBH ein.
Sie erhält den Status LOCKED, da sie noch nicht existiert.
Mit RECOVER CATALOG TO/USING REPLICATION wird die Datenbank aus dem Replikat erzeugt.

Folgende Punkte muss der Datenbankverwalter anschließend berücksichtigen:

- Er muss die Storage Groups in der neuen Datenbank ändern und anpassen.
- Er muss eigene SESAM-Sicherungsbestände mit COPY CATALOG und eventuell anschließendem COPY SPACE erzeugen, weil auf die alten Kopien nicht mehr zugegriffen werden kann. Er muss mit dem Utility-Monitor aus der CAT-REC-Datei die Metadaten zu alten SESAM-Sicherungsbeständen für den Catalog-Space entfernen. Mit MODIFY RECOVERY muss er alte Verweise auf SESAM-Sicherungsbestände für die Anwender-Spaces löschen.
- Wurde das neue Original aus einem Teilreplikat erstellt, müssen die fehlenden Spaces mit RECOVER... TO '*DUMMY' erzeugt oder explizit mit DROP SPACE FORCED gelöscht werden.

Zur weiteren Vorgehensweise siehe Abschnitt „Originaldatenbank mit einem Replikat reparieren“ auf Seite 256 und Abschnitt „Zurücksetzen der Originaldatenbank mit Hilfe eines Replikats“ auf Seite 260.

7.6.7 Replikat als Schattendatenbank nutzen

Als Schattendatenbank wird ein parallel zur Originaldatenbank geführtes Replikat dieser Datenbank bezeichnet. Mit einer Schattendatenbank können Sie:

– einen DBH im OLTP-Betrieb von rein lesenden Anwendungen entlasten
– eine Originaldatenbank reparieren
– eine Originaldatenbank zurücksetzen
– eine eigenständige Datenbank erzeugen

In diesem Abschnitt wird anhand eines Bildes beschrieben, wie Sie ein Replikat als Schattendatenbank halten. Erläuterungen zu den einzelnen Schritten finden Sie auf der nächsten Seite.

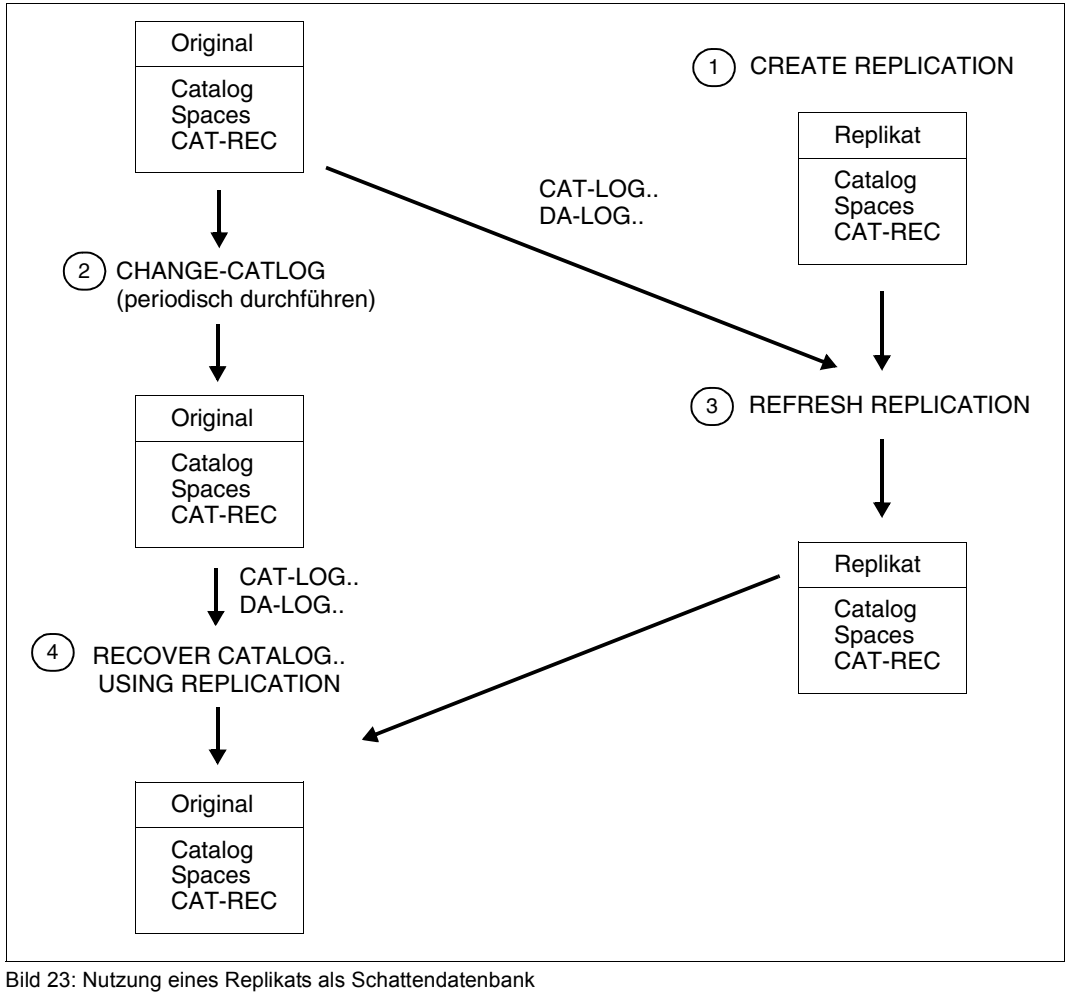


Bild 23: Nutzung eines Replikats als Schattendatenbank

1. Sie erzeugen ein Replikat mit CREATE REPLICATION aus einem SESAM-Sicherungsbestand oder einer Fremdkopie der Originaldatenbank.

   Um das Replikat als Schattendatenbank nutzen zu können, muss es regelmäßig aktualisiert werden. Dazu sind die Schritte 2 bis 4 notwendig:

2. In angemessenen Abständen wird ein Wechsel der Loggingdateien der Originaldatenbank veranlasst.
3. Mit der Utility-Anweisung REFRESH REPLICATION werden die Änderungen in den Loggingdateien im Replikat nachgefahren.

   Ist die Originaldatenbank defekt, soll sie mit Hilfe des Replikats repariert werden:

4. Mit der Utility-Anweisung RECOVER CATALOG..USING REPLICATION wird die Originaldatenbank repariert. Das Replikat dient dabei als Sicherungskopie. Dabei werden die Loggingdateien nachgefahren, die seit der letzten Aktualisierung des Replikats entstanden sind.

Sie können die Ausfallzeit der Originaldatenbank verkürzen, indem Sie den Parameter RENAME verwenden. In diesem Fall wird das Replikat umbenannt und damit zur Originaldatenbank. Während die Originaldatenbank wieder voll zur Verfügung steht, wird in einem weiteren Schritt ein neues Replikat erzeugt und erneut als Schattendatenbank eingesetzt.

Die Ausfallzeit können Sie noch weiter verkürzen, wenn Sie nur die defekten Spaces mit der Utiltiy-Anweisung RECOVER SPACE USING REPLICATION RENAME oder den Catalog-Space mit RECOVER CATALOG_SPACE USING REPLICATION RENAME reparieren. Im Replikat fehlen dann die Anwender-Spaces. Diese können mit REFRESH SPACE wieder in das Replikat aufgenommen werden. Alternativ können Sie auch das Replikat neu erzeugen.

7.7 Sicherungsspezifische Dateien und Catalog-Tabellen

Datei(en)	benutzt für	Zuweisung
TA-LOG1/2	Transaktionssicherung	576/48
DDL-TA-LOG	Transaktionssicherung bei DDL	1536/384
WA-LOG	Wiederanlauf	1320/48
DA-LOG	Logging für Anwender-Spaces	768/0
CAT-REC	Steuern des Recovery; es gibt sie als – Original – Kopie zum Aktualisieren eines Replikats – Replikat	12/12
CAT-LOG	Logging für Catalog-Space	768/0
PBI	physikalisches Before Image	576/72
Catalog-Tabelle RECOVERY_UNITS	Verwaltung der SESAM-Sicherungsbestände von Anwender-Spaces	
Catalog-Tabelle DA_LOGS	Verwaltung der DA-LOG-Dateien	

Tabelle 50: Sicherungsspezifische Dateien

Abweichend von den in der Tabelle angegebenen Größen der Dateizuweisung werden bei einem RECOVER die Dateien TA-LOG1/2 und WA-LOG stets in der Größe angelegt, die während der Bearbeitung des Original-Catalogs vorhanden war.

8 Datenbankbetrieb

Die zentrale Komponente zur Steuerung, Abwicklung und Überwachung des Datenbankbetriebs ist der Data Base Handler, kurz SESAM/SQL-DBH oder DBH. Ohne den DBH ist der Zugriff auf SESAM/SQL-Datenbanken nicht möglich.

Das Starten und Beenden des DBH sowie die Überwachung und Administration der laufenden DBH-Session sind Aufgaben des SESAM/SQL-Systemverwalters.

Dieses Kapitel bietet grundlegende Informationen zum Datenbankbetrieb. Es wendet sich besonders an die Leser, die sich für die SESAM/SQL-Systemverwaltung interessieren.

Die Abschnitte dieses Kapitels behandeln folgende Themen:

- Aufgaben und Einsatzmöglichkeiten des DBH
- Kommunikation zwischen Anwenderprogramm, DBH und Datenbank
- Zusammenfassung von Anwendungen in Konfigurationen
- Verteilte Verarbeitung mit SESAM/SQL-DCN
- Dateien des DBH
- Mechanismen zur Durchsatzsteigerung
- Überwachungs- und Steuerungsmöglichkeiten des Systemverwalters

8.1 Data Base Handler

8.1.1 Überblick

Ein SESAM/SQL-Anwenderprogramm kann auf die Daten einer Datenbank nicht direkt zugreifen. Der Zugriff auf die Daten einer Datenbank erfolgt vielmehr über den Data Base Handler (DBH). Zusammen mit den Konnektionsmodulen (siehe Seite 273) bildet der DBH die Schnittstelle zwischen Anwenderprogramm und Datenbank.

Der DBH führt folgende Aufgaben durch:

- er analysiert die Anweisungen aus den Anwenderprogrammen, ermittelt jeweils die optimale Zugriffsmethode und führt die Anweisungen aus
- er gibt die Ergebnisse von Datenbankoperationen zurück an das Anwenderprogramm
- er überwacht alle Datenbankaktivitäten
- er steuert die Service-Tasks für Utility-Funktionen
- er protokolliert wichtige Informationen zu Datenbankaktivitäten in DBH-spezifischen Protokolldateien (siehe Seite 302)
- er meldet aufgetretene Fehler.

Betrieb mit mehreren Datenbanken

Der SESAM/SQL-DBH unterstützt den Betrieb mit mehreren Datenbanken, wobei er bis zu 254 Datenbanken parallel bearbeiten kann. Somit kann auch jedes einzelne Anwenderprogramm gleichzeitig auf bis zu 254 Datenbanken zugreifen. Falls dies bei großen Anwendungen nicht ausreicht, ist der Einsatz der Verteilkomponente SESDCN zu erwägen (siehe Abschnitt „Verteilte Verarbeitung mit SESAM/SQL-DCN“ auf Seite 283).

Betrieb mit mehreren Auftraggebern

Mehrere Auftraggeber können gleichzeitig mit derselben SESAM/SQL-Datenbank arbeiten. Die Datenbankzugriffe der verschiedenen, parallel arbeitenden Auftraggeber verwaltet und koordiniert dabei der DBH.
SESAM/SQL lässt fast beliebig viele Auftraggeber zu. Als Auftraggeber gilt im Teilnehmerbetrieb ein Dialog- oder Batchprogramm, im Teilhaberbetrieb ein Terminal bzw. ein Paar, bestehend aus Benutzerkennung und Terminal.

8.1.2 Systemarchitektur

SESAM/SQL gibt es in 3 verschiedenen Produktvarianten:

- beim Independent DBH die Variante SESAM/SQL
  - als Standard Edition mit Singletask-Betrieb und
  - als Enterprise Edition, die den Multitask-Betrieb beinhaltet
- beim Linked-in DBH die Variante SESAM/SQL-LINK.

Bild 24 zeigt eine Gegenüberstellung von linked-in DBH und independent DBH.

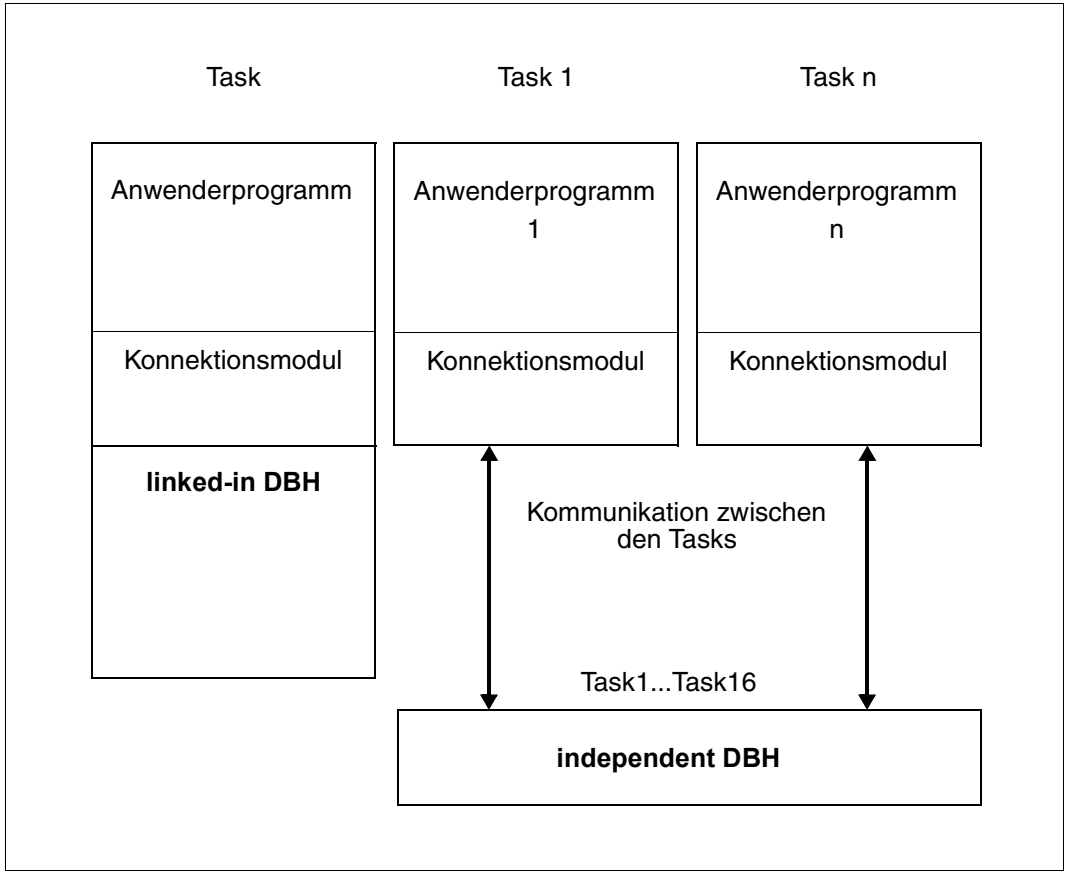


Bild 24: Gegenüberstellung von linked-in DBH und independent DBH

Independent DBH

Bild 25 zeigt den Independent DBH und seine Schnittstellen. Wie aus dem Bild hervorgeht, kann der Independent DBH auf mehrere Datenbanken zugreifen und mit mehreren Auftraggebern zusammenarbeiten.

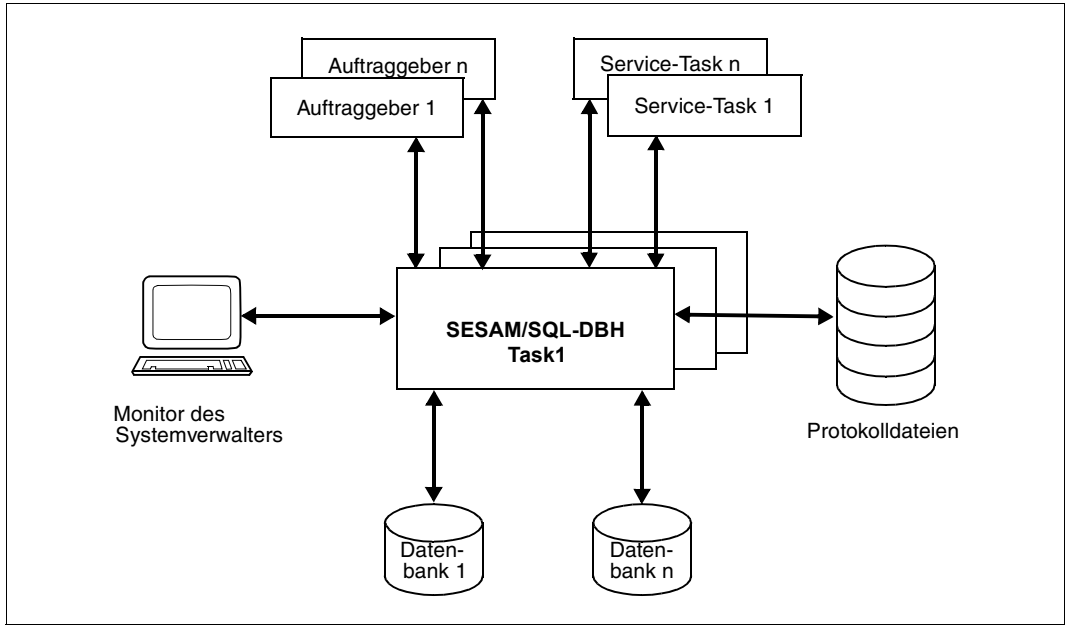


Bild 25: Der SESAM/SQL-DBH und seine Schnittstellen

Der **independent DBH** besteht aus einer oder mehreren Tasks (maximal 16) (Multitasking nur Enterprise Edition). Die Tasks dieser Task-Familie sind als Einheit zu verstehen, d.h. sie haben alle die gleiche Funktionalität und werden gemeinsam gestartet und beendet. Für den Systemverwalter bietet die Task-Familie ein „Single-System-Image“. Dies bedeutet:

- Er startet SESAM/SQL in einer Task, die wiederum entsprechend der Ladeoption DBH-TASKS den Start der weiteren zur Task-Familie gehörenden DBH-Tasks veranlasst. Der DBH ist erst betriebsbereit, wenn alle DBH-Tasks hochgefahren sind.
- Er kommuniziert immer nur mit der Task, die er explizit gestartet hat, aber die Administrationsanweisungen wirken immer auf alle Tasks. Eine einzelne Task der Task-Familie kann nicht gezielt angesprochen werden.
- Während der laufenden Session erfolgen alle Ausgaben von SESAM/SQL über die explizit gestartete DBH-Task.
- Die Lastverteilung auf die einzelnen DBH-Tasks nimmt SESAM/SQL eigenständig vor. Sie kann vom Systemverwalter nicht beeinflusst werden.

Linked-in DBH

Bild 26 zeigt den **linked-in DBH** und seine Schnittstellen. Wie aus dem Bild hervorgeht, kann der Linked-in DBH auf mehrere Datenbanken zugreifen, aber im Gegensatz zum Independent DBH nur mit einem Auftraggeber zusammenarbeiten. Er besteht immer aus einer Task.

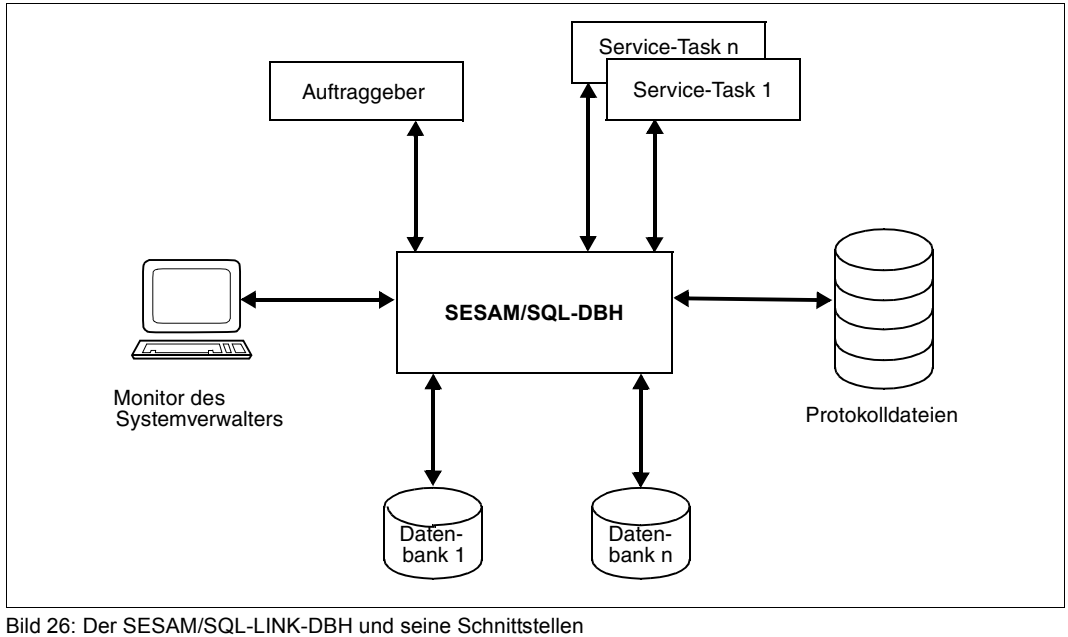


Bild 26: Der SESAM/SQL-LINK-DBH und seine Schnittstellen

Der linked-in DBH bearbeitet exklusiv die Aufträge eines einzigen Anwenderprogramms und wird direkt mit dem Anwenderprogramm gebunden. Bei Eingabe der ersten Anweisung lädt das Konnektionsmodul SESLINK automatisch die Module des linked-in DBH aus der SESAM/SQL-Modulbibliothek nach und fungiert während des Programmlaufs als Mittler zwischen Anwenderprogramm und DBH (siehe Abschnitt „Konnektionsmodule“ auf Seite 273).
Programm und DBH belegen eine gemeinsame Task. Dadurch entfallen beim Aufruf einer Datenbankoperation die Zeiten für die Kommunikation zwischen den Tasks. Da der linked-in DBH außerdem nur Anforderungen eines einzigen Anwenderprogramms verwalten muss, braucht er relativ wenig Zeit für die Ausführung einer Datenbankoperation. Nur bei hoher Update-Rate kann der linked-in DBH langsamer sein als der independent DBH, denn beim linked-in DBH wird stets nur ein Thread generiert (siehe Abschnitt „Multi-Thread-Betrieb“ auf Seite 317).
Das Arbeiten mit dem linked-in DBH setzt voraus, dass das Zusatzprodukt SESAM/SQL-LINK installiert ist. Sein Einsatz bietet sich immer dann an, wenn eine oder mehrere Datenbanken ausschließlich von einem Anwenderprogramm bearbeitet werden sollen.

8.1.3 Steuerungsmöglichkeiten des SESAM/SQL-Systemverwalters

Das Starten und Beenden des DBH, sowie die Betreuung der laufenden DBH-Session sind Aufgaben des SESAM/SQL-Systemverwalters. Die DBH-Session ist der Zeitraum zwischen Starten und normalem Beenden des DBH.

Über Einträge in SQL-Datenbankverzeichnissen bzw. CALL-DML-Tabellenverzeichnissen (siehe Handbuch „Datenbankbetrieb“) ordnet der Systemverwalter dem DBH die Datenbanken zu. Auf diese Datenbanken kann der DBH kann im Laufe der Session zugreifen.

Es gibt drei Möglichkeiten, die benötigten Datenbanken einzutragen:

1. über DBH-Startanweisungen beim Starten des DBH (siehe Seite 324)
2. über die Administrationsanweisungen ADD-SQL-DB-CATALOG-ENTRY bzw. ADD-OLD-TABLE-CATALOG-ENTRY während der laufenden Session (siehe Handbuch „Datenbankbetrieb“).
3. über die Utility-Anweisung CREATE CATALOG während der laufenden Session, wobei mit dieser Anweisung neu definierte Datenbanken eingetragen werden (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“).

Beim Starten des DBH kann der Systemverwalter den DBH außerdem über DBH-Optionen parametrisieren. Er kann Betriebsmittel, Grenzwerte und Arbeitsregeln den Anforderungen der aktuellen DBH-Session anpassen.

Der SESAM/SQL-Systemverwalter kann wichtige Informationen der DBH-Session auch automatisch per E-Mail versenden lassen, siehe Abschnitt „E-Mail-Versand wichtiger Informationen der DBH-Session“ auf Seite 335.

Auch während der laufenden DBH-Session kann der Systemverwalter im Rahmen der Administrationsanweisungen steuernd in den Betrieb eingreifen und die Einstellungen der DBH-Optionen an geänderte Bedingungen anpassen. Die aktuellen DBH-Optionen können in eine Datei gesichert und für die nächste DBH-Session verwendet werden.

Dem SESAM/SQL-Systemverwalter stehen die DBH-Administrationskommandos RECONFIGURE-DBH-SESSION und RELOAD-DBH-SESSION zur Steigerung der Verfügbarkeit des independent DBH, zum Einspielen einer Korrekturversion während des DBH-Betriebs und zur dynamischen Rekonfiguration der DBH-Session zur Verfügung (siehe Handbuch „Datenbankbetrieb“).
Beide Administrationsanweisungen werden ohne Unterbrechung der DBH-Session ausgeführt. Aus Anwendersicht kommt es nicht zu einem DBH-Ausfall.

8.2 Konnektionsmodule

Jedes Anwenderprogramm, das SESAM/SQL-Datenbanken bearbeiten soll, muss mit einem Konnektionsmodul gebunden werden. Das Konnektionsmodul baut die Kommunikation mit dem SESAM/SQL-DBH auf und wickelt den Datentransfer zwischen Anwenderprogramm und DBH ab.

In TIAM- und DCAM-Anwendungen übergibt das Anwenderprogramm seine Anweisungen per Unterprogrammaufruf direkt an das Konnektionsmodul. Dieses setzt die Anweisungen um und leitet sie an den DBH weiter. Der DBH führt die Anweisungen aus und überträgt die Ergebnisse von Datenbankoperationen zurück an das Konnektionsmodul, das sie seinerseits an das Anwenderprogramm weiterleitet.

Jedes Konnektionsmodul kann sowohl von XS-fähigen als auch von nicht-XS-fähigen Anwendungen verwendet werden. Innerhalb eines Programms ist es sogar möglich, den Adressierungsmodus zwischen zwei Aufrufen des Konnektionsmoduls zu wechseln.

Ein Teil eines Anwenderprogramms kann im XS-Modus, ein anderer im Nicht-XS-Modus ablaufen. In beiden Teilen können Datenbankaufrufe eingegeben werden.

In linked-in Anwendungen dürfen SQL- und CALL-DML-Anweisungen nicht gemischt vorkommen.

8.2.1 Konnektionsmodul-Varianten

Welches Konnektionsmodul zu einem Anwenderprogramm gebunden werden muss, hängt davon ab, ob das Anwenderprogramm mit dem independent DBH oder dem linked-in DBH zusammenarbeiten soll.

Für Anwenderprogramme, die mit dem independent DBH zusammenarbeiten, existiert pro Betriebsart genau ein Konnektionsmodul:
TIAM-Anwendungen werden mit dem Konnektionsmodul SESMOD gebunden,
DCAM-Anwendungen mit SESDCAM und UTM-Anwendungen mit SESUTMC.

Anwenderprogramme, die mit dem linked-in DBH zusammenarbeiten, müssen stets mit dem Konnektionsmodul SESLINK gebunden werden. Bei den Dienstprogrammen SEDI61L und SEDI63L ist das Modul SESLINK bereits eingebunden.

Konnektionsmodul	**DBH**	**Betriebsart**	**Oberbegriff**
SESMOD	independent DBH	TIAM	DBCON
SESDCAM	independent DBH	DCAM (nur CALL-DML)	
SESUTMC	independent DBH	UTM	
SESLINK	linked-in DBH		DBCONL

Tabelle 51: Konnektionsmodul-Varianten

8.2.2 Konnektionsmodul-Parameter

Zur Erfüllung seiner Funktionen benötigt DBCON verschiedene Parameter. Wichtige Konnektionsmodul-Parameter sind der DBH-Name und der Konfigurationsname. Diese Parameter ordnen das Anwenderprogramm, das mit dem Konnektionsmodul gebunden wird, einem DBH und einer Konfiguration zu. Daneben gibt es noch einige weitere Parameter zur Konfigurierung von DBCON.

TIAM-, DCAM- und UTM-Anwendungen

Konnektionsmodule für TIAM-, DCAM- und UTM-Anwendungen, die mit dem independent DBH zusammenarbeiten (DBCON), werden normalerweise über die Konfigurationsdatei des Anwenderprogramms parametrisiert (siehe Abschnitt „Konfigurationsdatei“ auf Seite 296).

Der Anwender trägt die Konnektionsmodul-Parameter in die Konfigurationsdatei ein und weist diese vor dem Starten des mit dem Konnektionsmodul gebundenen Anwenderprogramms zu. Im Laufe der Startprozedur werden die eingetragenen Parameter dann an das Konnektionsmodul übergeben.

Im linked-in Fall, in dem der DBH direkt mit dem Anwenderprogramm gebunden wird, sind DBH-Name und Konfigurationsname als DBH-Startparameter in der Konfigurationsdatei des DBH enthalten (siehe Abschnitt „Konfigurationsdatei für DBH-Startparameter“ auf Seite 297). Für DBCONL gibt es keine Konfigurierungsparameter.

Im Folgenden sind die Konnektionsmodul-Parameter für TIAM-, DCAM- und UTM-Anwendungen jeweils alphabetisch aufgelistet:

- Zunächst finden Sie eine Zusammenstellung solcher Konnektionsmodul-Parameter, die für TIAM-, DCAM- und UTM-Anwendungen gelten.
- Anschließend sind Konnektionsmodul-Parameter aufgelistet, die nur für DCAM- und UTM-Anwendungen bzw. nur für DCAM-Anwendungen gelten. Konnektionsmodul-Parameter, die nur für UTM-Anwendungen gelten, finden Sie im Abschnitt „Starten einer SESAM/SQL-UTM-Anwendung“ auf Seite 403.

Alle Parameter sind DBCON-Parameter.

Konnektionsmodul-Parameter für TIAM-, DCAM- und UTM-Anwendungen

In den Parameterzeilen sind Leerzeichen **nicht** erlaubt.

CCSN=*ccs-name* Codierter Zeichensatz, mit dem das Anwenderprogramm arbeitet.

ccs-name: Name des codierten Zeichensatzes, so wie er im BS2000 (Systemkomponente XHCS) definiert ist bzw. *NONE, wenn für das Anwenderprogramm kein codierter Zeichensatz angegeben werden soll. Standardwert: *NONE

Wenn für die Datenbank ein codierter Zeichensatz verwendet wird (Klausel CODE-TABLE *ccs_name* in den Anweisungen CREATE CATALOG oder ALTER CATALOG), dann muss, bei Zugriffen des Anwenderprogrammes auf die Datenbank, der CCS-Name der Datenbank mit dem CCS-Namen des Anwenderprogrammes übereinstimmen. Zugriffe von Anwenderprogrammen mit CCSN=*NONE bzw. von Anwenderprogrammen aus SESAM/SQL < V5.0 werden mit SQLSTATE abgewiesen.

Für die Utility-Anweisungen CREATE/ALTER CATALOG, CREATE/REFRESH REPLICATION und RECOVER CATALOG [SPACE] findet diese Prüfung nicht statt.
Wenn für die Datenbank kein CCSN angegeben wurde, dann findet diese Prüfung ebenfalls nicht statt.

CNF=*k* Name der Konfiguration, in der das Anwenderprogramm arbeiten soll

k: A..Z, 0..9, ␣
Standardwert: ␣

DIAG-DUMP=(*diag*) Kriterium für die Erstellung eines Diagnose-Dump.

diag: {SQLSTATE=*state*|STATUS=*status*}

state: SQLSTATE mit Class und Subclass (5 Byte);
status: CALL-DML-Status mit Unternummer (4 Byte)

SQLSTATE kann teilqualifiziert angegeben werden durch Eingabe von „***“ als Subclass.
CALL-DML-Status kann teilqualifiziert angegeben werden durch Eingabe von „**“ als Unternummer.

Beim ersten Auftreten der entsprechenden Fehlermeldung (SQLSTATE bzw. CALL-DML-Status) wird ein Dump der Anwendertask erstellt (siehe Handbuch „Datenbankbetrieb“)

ISOL-LEVEL=*level* Standardwert für den Isolationslevel von SQL-Transaktionen. Nur wirksam in SQL-Anwendungen.

level: {READ-UNCOMMITTED | READ-COMMITTED | REPEATABLE-READ | SERIALIZABLE}
Standardwert: SERIALIZABLE

Gültig für alle Transaktionen des ESQL-Anwenderprogramms, wenn der Isolationslevel nicht durch die entsprechende SQL-Anweisung im Programm festgelegt ist.

Dieser Parameter wird nur ausgewertet von TIAM- und UTM-Anwendungen.

NAM=*x* Name des DBH, mit dem das Anwenderprogramm zusammenarbeiten soll

x: A..Z, 0..9, ␣
Standardwert: ␣

NOTYPE Meldung `SESAMxx not available ...` wird unterdrückt

NVT Die Anwendung kann nur lokal arbeiten und nur mit dem DBH kommunizieren, der beim Parameter NAM eingetragen wird. Dies gilt auch, wenn SESDCN geladen ist.

PREFETCH-BUFFER=*puffergröße*
Größe des Speicherbereichs, der für den Prefetch verwendet wird, in Kbyte.
Wertebereich: 0 ≤ puffergröße ≤ 4096

Dieser Parameter wird nur von TIAM- und UTM-Anwendungen ausgewertet.

PRIO-CHECK=*t* Zeitintervall zwischen zwei aufeinanderfolgenden Prüfungen der BS2000-Task-Priorität des Anwenderprogramms in Sekunden.
$10 \leq t \leq 3600$
Standardwert: 300

PRIO-CHECK=OFF Prüfung der BS2000-Task-Priorität des Anwenderprogramms wird nach der ersten Prüfung abgeschaltet.

SQLOPT-KEEP-JOIN-ORDER={YES | NO}
Standardwert: NO

NO Der SESAM/SQL-Optimizer entscheidet über die die Reihenfolge der Join-Operanden bei mehrfachen Joins.

YES Der SESAM/SQL-Optimizer hält sich bei der Join-Reihenfolge soweit möglich an die in der SQL-Anweisung vorgegebene Reihenfolge der Join-Operanden. Dies gilt für explizite Joins (z.B. *table_1* JOIN *table_2* ON *suchbedingung*).

SQLOPT-PREFERRED-JOIN-METHOD={SORT-MERGE | NESTED-LOOP | NONE}
Standardwert: NONE

NONE Der SESAM/SQL-Optimizer entscheidet über den Join-Algorithmus.

SORT-MERGE
NESTED_LOOP Der SESAM/SQL-Optimizer verwendet, sofern möglich, den angegebenen Join-Algorithmus.

Verwenden Sie diese beiden SQL-Hinweise mit Bedacht! Sie gelten für alle Anwendungen, die mit dieser Konfigurationsdatei arbeiten.
Pragmas und Annotationen der SQL-Anweisungen haben aber Vorrang vor diesen SQL-Hinweisen der Konfigurationsdatei.

TRACE,TYPE=*type* Protokollierung des Call- bzw. Message-Trace ab Start des Anwenderprogramms einschalten.

type: {CALL|MSG | (CALL,MSG)}
[,OUTPUT={SYSOUT | SYSLST | (SYSOUT,SYSLST)}]
Standardwert für Ausgabe (OUTPUT): SYSLST

Weitere Informationen zum Call- und Message-Trace siehe Handbuch „Datenbankbetrieb“.

DCAM- und UTM-spezifische Konnektionsmodul-Parameter

PUF=*p* maximale Nachrichtenlänge bei DCAM- bzw. UTM-Anwendungen in Byte
$1 \le p \le 64000$
Standardwert: 4096

Eine Nachricht kann sein:
– die Anforderung (Anweisung) eines Anwenderprogramms an den SESAM/SQL-DBH
– die Antwort des SESAM/SQL-DBH an das Anwenderprogramm

Da die Länge von SQL-Nachrichten nur schwer einschätzbar ist, empfiehlt es sich, bei Anwendungen mit SQL-Anweisungen für *p* den Wert 64000 zu wählen.

TOTAL-APPL=*a* maximale Anzahl SESAM/SQL-Teilhaber-Anwendungen in der Konfiguration (nur relevant für Anwendungen, die nicht verteilt arbeiten)
$1 \le a \le 128$
Standardwert: 64

TOTAL-USERS=*u* maximale Anzahl SESAM/SQL-Teilhaber in der Konfiguration (nur relevant für Anwendungen, die nicht verteilt arbeiten)
$1 \le u \le 16000$
Standardwert: 128

Die Parameter TOTAL-APP und TOTAL-USERS gelten für die gesamte Konfiguration. Die zuerst gestartete Applikation legt auf Grund dieser Parameter die Größe des Teilhaberpools fest. Später gestartete Anwendungen klinken sich nur noch an den Pool an. Die dort angegebenen Parameter werden ignoriert.

DCAM-spezifische Konnektionsmodul-Parameter

NAME-APPL=*appl* Applikationsname der DCAM-Anwendung

appl: 1 bis 8 beliebige abdruckbare Zeichen
Standardwert: TSN=tsn

REQUEST-USERS=*r*
maximale Anzahl paralleler asynchroner Aufrufe pro Task bei DCAM-Anwendungen
$1 \le r \le 128$
Standardwert: 7

START Kaltstart bei DCAM-Anwendungen
Standardwert: Warmstart

8.3 Zusammenfassung von Anwendungen in Konfigurationen

In einem Rechner können gleichzeitig mehrere SESAM/SQL-Anwendungen ablaufen, an denen jeweils ein DBH mit seinen dazugehörigen Anwenderprogrammen beteiligt ist.

Bei verteilter Verarbeitung, bei der Anwenderprogramme mit mehr als einem DBH zusammenarbeiten, können verschiedene DBHs, Anwenderprogramme und Verteilkomponenten (SESDCNs) an Anwendungen beteiligt sein (siehe Abschnitt „Verteilte Verarbeitung mit SESAM/SQL-DCN“ auf Seite 283).

Um Konflikte zu vermeiden, kann der Systemverwalter zusammengehörige Anwendungen in Konfigurationen zusammenfassen und so gegenüber anderen Konfigurationen abschotten. So ist es z.B. möglich, Test- und Produktivanwendungen jeweils in eigenen Konfigurationen zusammenzufassen, so dass sie völlig unabhängig voneinander arbeiten können.

Eine Konfiguration kann folgende Komponenten enthalten:

- bei nicht-verteilter Verarbeitung: einen oder mehrere DBHs mit den zugeordneten Datenbanken sowie ein oder mehrere Anwenderprogramm(e)
- bei verteilter Verarbeitung: alle zusammengehörigen DBHs (jeweils mit den zugeordneten Datenbanken), Anwenderprogramme und SESDCNs
- zusätzlich definiert SESAM/SQL für jede Konfiguration mit mindestens einer Teilhaberanwendung sowie für jede verteilt arbeitende Konfiguration einen Pool zur Aufnahme der Verwaltungsdaten der Kommunikation bzw. der verteilten Verarbeitung.

Alle Komponenten, die zu einer Konfiguration zusammengefasst werden sollen, müssen auf demselben Rechner liegen. Jeder DBH und jedes Anwenderprogramm, bei verteilter Verarbeitung auch jeder SESDCN, muss zum Ablaufzeitpunkt genau einer Konfiguration zugeordnet sein.

Konfigurationsname

Der Konfigurationsname ordnet die einzelnen Komponenten ihrer jeweiligen Konfiguration zu. Das Anwenderprogramm erfährt den Konfigurationsnamen über einen Konnektionsmodul-Parameter in der Konfigurationsdatei (siehe Seite 274), der DBH und SESDCN erfahren den Konfigurationsnamen über eine DBH- bzw. DCN-Option (siehe Seite 325 und Seite 328).

Innerhalb des Rechners muss der Konfigurationsname eindeutig sein. Konfigurationen, die auf verschiedenen Rechnern liegen, können gleichnamig sein. Es ist aber empfehlenswert, auch rechnerübergreifend verschiedene Konfigurationsnamen zu vergeben, damit ggf. ein SESDCN-Wiederanlauf auf einem anderen als dem Kaltstartrechner möglich ist (siehe Handbuch „Datenbankbetrieb“).

Kommunikation zwischen Konfigurationen

Die einzelnen Konfigurationen arbeiten völlig unabhängig voneinander. Ohne die Verteilkomponente SESDCN kann ein Anwenderprogramm nur mit einem DBH kommunizieren, nämlich mit demjenigen, dessen Name als Konnektionsmodul-Parameter zugewiesen wurde, und der zur selben Konfiguration gehört wie das Anwenderprogramm. Mit SESDCN arbeiten die Konfigurationen zwar unabhängig voneinander, Anwenderprogramme können aber auch Anforderungen an DBHs eingeben, die nicht der eigenen Konfiguration angehören, selbst wenn diese auf anderen Rechnern liegen.

Bild 27 auf der folgenden Seite zeigt als Beispiel zwei Rechner mit je zwei Konfigurationen:

- Konfiguration M auf Rechner 1 besteht aus Anwenderprogrammen, SESDCNA, DBH1 DBH2 und dem Pool. Sie arbeitet unabhängig von Konfiguration Z im eigenen Rechner, kann aber über Rechnergrenzen hinweg mit Konfiguration J und Konfiguration Y kommunizieren.
- Konfiguration Z auf Rechner 1 besteht aus Anwenderprogrammen, DBH3 und dem Pool. Sie kommuniziert mit keiner anderen Konfiguration.
- Konfiguration J auf Rechner 2 besteht aus Anwenderprogrammen, SESDCNX, DBH6 und dem Pool. Sie kommuniziert sowohl mit Konfiguration Y auf demselben Rechner, als auch mit Konfiguration M auf Rechner 1.
- Konfiguration Y auf Rechner 2 besteht aus Anwenderprogrammen, SESDCNB, DBH7 DBH8 und dem Pool. Sie kommuniziert mit Konfiguration J auf demselben Rechner und mit Konfiguration M auf Rechner 1.

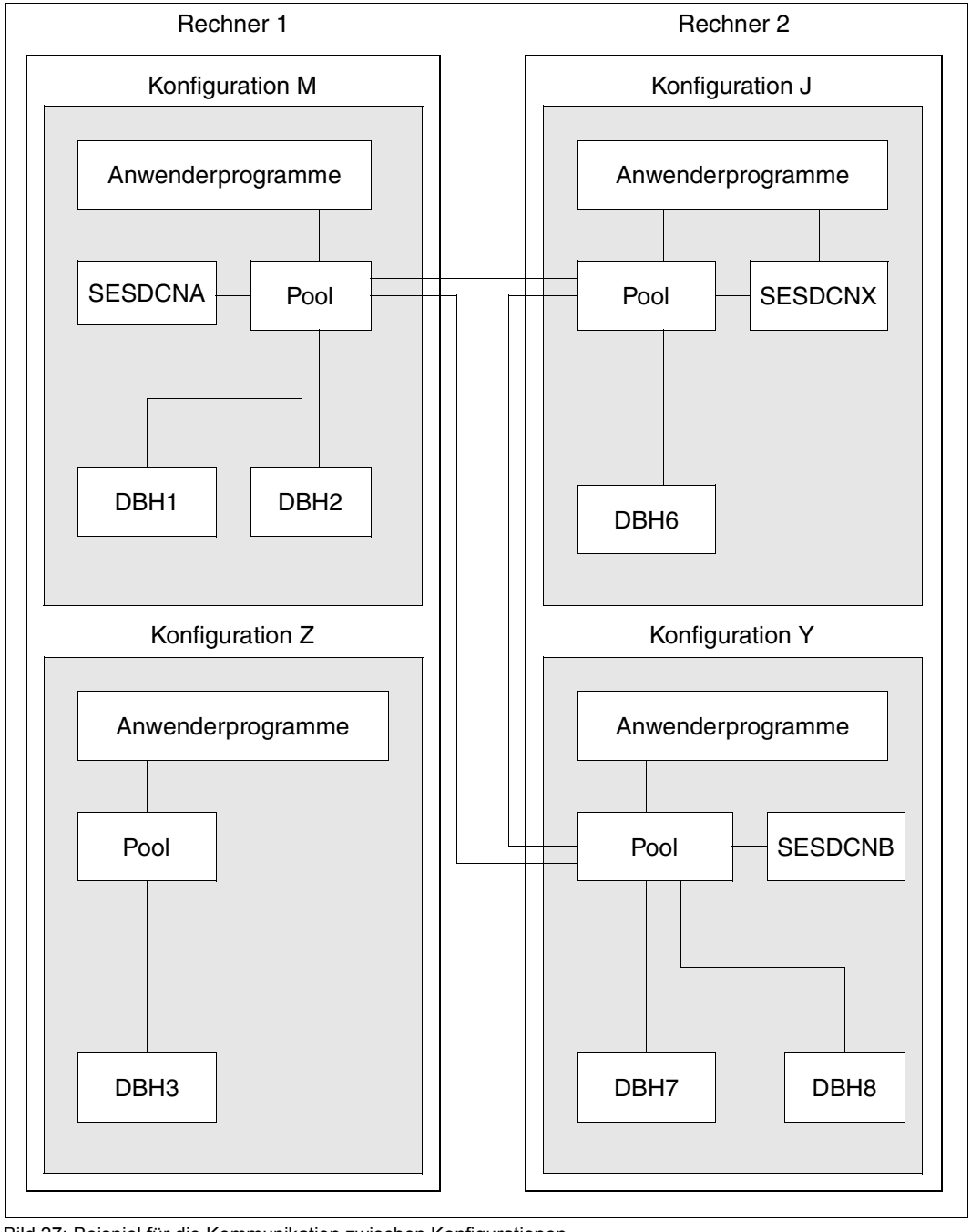


Bild 27: Beispiel für die Kommunikation zwischen Konfigurationen

Kommunikation mit virtuellen Hosts

Beim Betrieb einer SESAM-Konfiguration kann auch ein virtueller Host genutzt werden. SESAM/SQL benutzt dann nicht den realen, sondern den virtuellen Host als Kommunikationspartner.

Dadurch können Sie Verteilregeln und SQL-Systemzugänge unabhängig von realen Hostnamen machen.

Damit SESAM/SQL einen virtuellen Host nutzen kann, muss in der Datei $TSOS.SYSDAT.BCAM.APPLICATIONS eine entsprechende Zuweisung für die Applikation SES090*cnf* eingetragen werden (*cnf* ist dabei der Konfigurationsname).

Beispiel

Eintrag mit *cnf* = *P* in der Datei $TSOS.SYSDAT.BCAM.APPLICATIONS:

```
NEA     SES090P     VIRHOST
```

Als Folge nutzt SESAM/SQL bei der Auftraggeber-Identifikation (<user-identifikation>) als Hostnamen den virtuellen Hostnamen. Ohne diese Zuweisung erfolgt die Auftraggeberidentifikation wie bisher über den realen Hostnamen.

Wenn der Eintrag in der Datei $TSOS.SYSDAT.BCAM.APPLICATIONS erfolgt ist, dann darf die Zuordnungsdatei (von Anwendungen zu virtuellen Hosts) nicht dynamisch verändert werden (siehe Parameter APPLICATION-TABLE in den BCAM-Kommandos DCSTART und DCOPT).

Bei TIAM- und DCAM-Anwendungen erfolgt die Nutzung des virtuellen Hosts automatisch.

Bei UTM-Anwendungen muss bei der UTM-Generierung in der MAX-Option der Parameter HOSTNAME mit dem Namen des virtuellen Hosts versorgt werden. Damit können auch die UTM-Anwendungen den virtuellen Host nutzen und mit der SESAM-Konfiguration kommunizieren. Dies ist zwingend notwendig, da alle Teilnehmer einer Konfiguration, wie bei realen Hosts, den gleichen Host nutzen müssen.

Der virtuelle Host darf nicht deaktiviert werden, wenn auf ihm eine SESAM-Konfiguration betrieben wird, da dies als Ausfall des Kommunikationspartners erkannt wird, selbst wenn der reale Host noch aktiv ist.

8.4 Verteilte Verarbeitung mit SESAM/SQL-DCN

Bei der verteilten Verarbeitung mit SESAM/SQL-DCN kann ein Anwenderprogramm innerhalb einer Session mit mehr als einem SESAM/SQL-DBH zusammenarbeiten. Diese Zusammenarbeit zwischen Anwenderprogramm und den verschiedenen DBHs kann innerhalb derselben Konfiguration, konfigurationsübergreifend innerhalb desselben Rechners oder auch rechnerübergreifend erfolgen. Die rechnerübergreifende Zusammenarbeit zwischen Anwenderprogramm und unterschiedlichen DBHs ermöglicht den Zugriff eines Anwenderprogramms auf Datenbanken, die auf unterschiedlichen Rechnern liegen. Man spricht dann von verteilten Datenbanken.

Bei verteilter Verarbeitung werden die Verarbeitungsrechner als Home- bzw. Remote-Rechner bezeichnet. Dabei beziehen sich die Ausdrücke „Home“ und „Remote“ auf die Sicht des Anwenderprogramms.

Home-Rechner

Der Rechner, in dem das Anwenderprogramm abläuft, wird als Home-Rechner bezeichnet.

Remote-Rechner

Innerhalb eines Rechnernetzes werden bzgl. eines Anwender-Programms alle Rechner außer dem Home-Rechner als Remote-Rechner bezeichnet.

8.4.1 Einsatzmöglichkeiten der verteilten Verarbeitung

Folgende Einsatzmöglichkeiten ergeben sich für die verteilte Verarbeitung mit SESAM/SQL-DCN:

- Bei verteilten Datenbanken ist es möglich, innerhalb einer Session Datenbanken auf verschiedenen Rechnern zu bearbeiten. Die Daten lassen sich dann jeweils auf dem Rechner speichern, auf dem sie am häufigsten benötigt werden. Das Verteilgranulat ist die SESAM/SQL-Datenbank.
- Konfigurationen können miteinander kommunizieren (innerhalb eines Rechners und über Rechnergrenzen hinweg).

Die verteilte Verarbeitung mit SESAM/SQL-DCN zeichnet sich durch folgende Eigenschaften aus:

- Verteilungsunabhängigkeit der Anwenderprogramme:
  Für die Anwenderprogramme ist der Ort einer Datenbank nicht relevant. Er wird intern vom System ermittelt und von den Anwenderprogrammen nicht explizit angesprochen. Die Anwenderprogramme sind deshalb von einer Änderung der Verteilung nicht berührt. Insbesondere können Anwenderprogramme, die bisher im nicht-verteilten Betrieb verwendet wurden, unverändert für die Bearbeitung im verteilten Betrieb übernommen werden. Die Verteilung der Datenbanken ist also für das Anwenderprogramm transparent, d.h. durchsichtig.
- Two-Phase-Commit bei Transaktionsende:
  Eine rechnerübergreifende Transaktionssicherung gewährleistet die transaktionsgesicherte netzweite Bearbeitung der Daten. Dies ermöglicht Änderungen in mehreren Datenbanken innerhalb einer Transaktion sowie den automatischen Wiederanlauf nach Systemausfall (siehe Abschnitt „Wiederanlauf“ auf Seite 207).
- Deadlock- und Longlock-Erkennung:
  Deadlock- und Longlock-Situationen werden vom System über Rechnergrenzen hinweg erkannt und aufgelöst.

Die Vorteile verteilter Datenbanken sind:

- Höhere Leistungsfähigkeit:
  Durch das Abarbeiten der Anwenderaufträge auf verschiedenen Rechnern kann unter Umständen eine Durchsatzsteigerung erzielt werden.
- Höhere Verfügbarkeit:
  Mehrere, auch unabhängig voneinander einsatzfähige DV-Anlagen stellen sicher, dass sich der Ausfall eines Rechners nicht auf alle DV-Verfahren auswirkt.
- Anpassungsfähige Organisation:
  Die Arbeitsabläufe müssen sich nicht mehr an einem zentralen Rechenzentrum orientieren.
- Flexibler Kapazitätsausbau

Betriebsarten einer SESAM/SQL-DCN-Anwendung

Die verteilte Verarbeitung mit SESAM/SQL-DCN ist sowohl im Teilnehmerbetrieb (TIAM) als auch im Teilhaberbetrieb möglich. Der Teilhaberbetrieb kann realisiert werden durch den Einsatz des Universellen Transaktions-Monitors openUTM oder des Datenkommunikationssystems DCAM.

8.4.2 Verteilkomponente SESDCN

Die Verteilkomponente SESDCN ist zentraler Bestandteil des Produkts SESAM/SQL-DCN. Die Verteilkomponente SESDCN muss in allen Konfigurationen geladen sein, die an der verteilten Verarbeitung teilnehmen. Die Konfigurationen können dabei alle auf demselben Rechner oder auch auf verschiedenen Rechnern liegen. Greift ein Anwenderprogramm auf eine Datenbank zu, die sich unter der Kontrolle eines DBH befindet, der einer anderen Konfiguration angehört als das Anwenderprogramm, so spricht man von einem Remote-Zugriff. Im Gegensatz dazu spricht man von einem lokalen Zugriff, wenn ein Anwenderprogramm nur auf Datenbanken zugreift, die sich unter der Kontrolle eines DBH befinden, der derselben Konfiguration angehört wie das Anwenderprogramm.

Die Aufgaben der Verteilkomponente SESDCN sind im Einzelnen:

- Aufbau und Verwaltung der Verteilregeln
- Entgegennahme von Remote-Zugriffen und Weiterleiten an den zuständigen SESAM/SQL-DBH
- Administration des Datenbankbetriebs (z.B. Ausführen von Administrationsanweisungen, Erkennen von Sperren)
- Überwachen von lokalen DBHs und Anwenderprogrammen
- Koordination und Durchführung des SESDCN-Wiederanlaufs
- Deadlock-Erkennung

Distributed Data Base Handler

In jeder Konfiguration bzw. jedem Rechner, in dem ein Anwenderprogramm auf eine oder mehrere Datenbanken zugreift, muss ein SESAM/SQL-DBH geladen sein. Der DBH steuert und verwaltet die Datenbankzugriffe. Die Gesamtheit aller DBHs, die an der verteilten Verarbeitung beteiligt sind, heißt Distributed Data Base Handler, kurz DDBH.

Verteilung von Anweisungen

Bei der verteilten Verarbeitung mit SESAM/SQL-DCN schickt das Konnektionsmodul DBCON die SQL- und/oder CALL-DML-Anweisungen, die ein Anwenderprogramm innerhalb einer Session absetzt, zur Bearbeitung an unterschiedliche DBHs.

Jede Anweisung, die keinen Vorgang und keine Transaktion beendet, schickt das Konnektionsmodul vollständig an einen bestimmten DBH. Eine Anweisung wird also nur von einem DBH bearbeitet. Dieses Verhalten heißt „function shipping“. Nur solche Anweisungen, die die aktuelle Transaktion beenden (COMMIT, ROLLBACK bei SQL, ETA bei CALL-DML), werden an alle DBHs geschickt, die an der aktuellen Transaktion beteiligt sind.

Definition der Verteilung in den Verteilregeln

Bei der Verteilung von Anweisungen bleibt es dem Anwenderprogramm verborgen, mit welchem DBH es gerade zusammenarbeitet. Auch bei der verteilten Verarbeitung muss ein Anwenderprogramm somit nur die Datenbank „kennen“, in der es z.B. eine bestimmte Tabelle anspricht.

Nicht zu wissen braucht das Anwenderprogramm hingegen,

- auf welchem Rechner die Datenbank liegt,
- welcher Konfiguration die Datenbank zugeordnet ist,
- an welche Verteilkomponente das Konnektionsmodul DBCON die Anweisungen schickt (nur bei Remote-Zugriffen),
- welcher DBH die Anweisungen bearbeitet.

Diese Angaben ermittelt das Konnektionsmodul selbstständig aus den Verteilregeln, die für jede Datenbank den zugehörigen „Zugriffspfad“, bestehend aus Rechnername, Konfigurationsname, DCN-Name und DBH-Name festlegen. Da das Konnektionsmodul den Datenbanknamen als Suchkriterium verwendet, muss der Datenbankname innerhalb der verteilten Anwendung, d.h. netzweit eindeutig sein.

Die Definition und Administration der Verteilregeln erfolgt über die Verteilkomponente SESDCN. Zur Definition der Verteilregeln verwendet der Systemverwalter die SESDCN-Steueranweisungen ADD-DISTRIBUTION-RULE-LIST und ADD-NETWORK-LINK-LIST. Aufgrund dieser Anweisungen baut SESDCN die Verteilregeln auf. Mit Hilfe von Administrationsanweisungen kann der Systemverwalter darüber hinaus die Verteilregeln im laufenden Betrieb den aktuellen Erfordernissen anpassen. SESDCN wird über dieselben Schnittstellen administriert wie der DBH.

Durchsatzsteigerung durch Laden mehrerer SESDCNs

In den meisten Fällen ist es ausreichend, pro Konfiguration SESDCN nur einmal zu laden. Bei Laufzeitverschlechterungen auf Grund vieler Remote-Zugriffe an einen SESDCN lässt sich jedoch der Gesamtdurchsatz bei Remote-Zugriffen durch das Laden mehrerer SESDCNs steigern. Durch Parallelbearbeitung der Remote-Aufträge wird so eine bessere Auslastung der Übertragungskomponenten erreicht.
Beim Laden mehrerer SESDCNs ist zu beachten, dass einem DBH immer nur ein SESDCN zugeordnet werden kann.

8.4.3 Zusammenwirken von SESAM/SQL-DBH und SESAM/SQL-DCN

Die verteilte Verarbeitung mit SESAM/SQL-DCN kann konfigurationsübergreifend innerhalb eines Rechners oder auch rechnerübergreifend erfolgen. Die Kommunikation zwischen Anwenderprogramm, SESDCN und Distributed Data Base Handler läuft in beiden Fällen nach demselben Muster ab, das in Bild 28 veranschaulicht ist. Die Konfigurationen A und B in Bild 28 kann man sich somit auf demselben oder auf verschiedenen Rechnern liegend vorstellen.

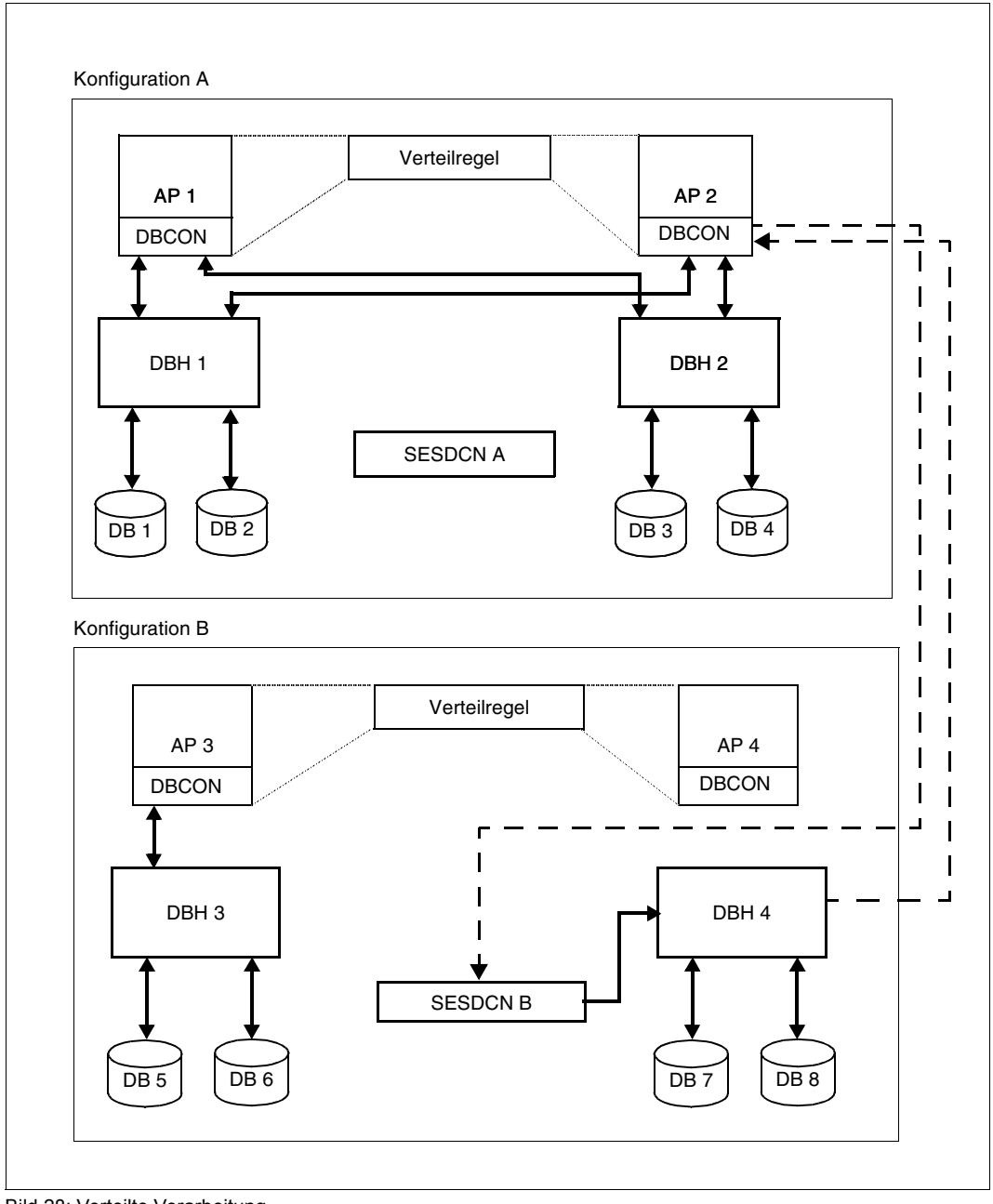


Bild 28: Verteilte Verarbeitung

Jedes Anwenderprogramm muss mit einem Konnektionsmodul DBCON gebunden werden. Das Konnektionsmodul stellt die Verbindung zwischen dem Anwenderprogramm und dem DBH her, der für die Bearbeitung der gewünschten Datenbank zuständig ist.

Zwei Fälle sind zu unterscheiden:

- Lokaler Zugriff (Anwenderprogramm und Datenbank gehören derselben Konfiguration an):
  In diesem Fall schickt das Konnektionsmodul die Anweisung an den laut Verteilregeln zuständigen DBH. Im oberen Teil von Bild 28 trifft dies beispielsweise für das Anwenderprogramm AP 1 und die Datenbank DB 3 zu. Der lokale Zugriff ist durch eine durchgezogene Linie dargestellt.
- Remote-Zugriff (Anwenderprogramm und Datenbank gehören verschiedenen Konfigurationen an):
  In diesem Fall schickt das Konnektionsmodul die Anweisung an die laut Verteilregeln zuständige Verteilkomponente SESDCN. Gleichzeitig teilt das Konnektionsmodul der Verteilkomponente den laut Verteilregeln für die Bearbeitung der Datenbank zuständigen DBH mit. Die Verteilkomponente schickt nun ihrerseits die Anweisung an den betreffenden DBH. Nach Ausführung der Anweisung schickt der DBH die Antwort direkt zurück an das Konnektionsmodul des Anwenderprogramms, das den Auftrag erteilt hat.
  In Bild 28 trifft die geschilderte Situation beispielsweise für das Anwendungsprogramm AP 2 und die Datenbank DB 8 zu. Der Remote-Zugriff ist durch eine gestrichelte Linie dargestellt.

8.4.4 Datensicherheit bei verteilter Verarbeitung

Der vorliegende Abschnitt erläutert Aspekte des Sicherungskonzepts, die speziell bei verteilter Verarbeitung mit SESAM/SQL-DCN von Bedeutung sind. Die konfigurationsübergreifende Transaktionssicherung stellt sicher, dass auch nach Ausfall einer Komponente des Rechnernetzes keine Datenverluste und Inkonsistenzen auftreten können.

Verteilte Transaktionen

Eine Transaktion heißt verteilt, wenn mehrere DBHs an ihrer Bearbeitung beteiligt sind. Eine verteilte Transaktion wird in drei Schritten abgewickelt (siehe Bild 29).

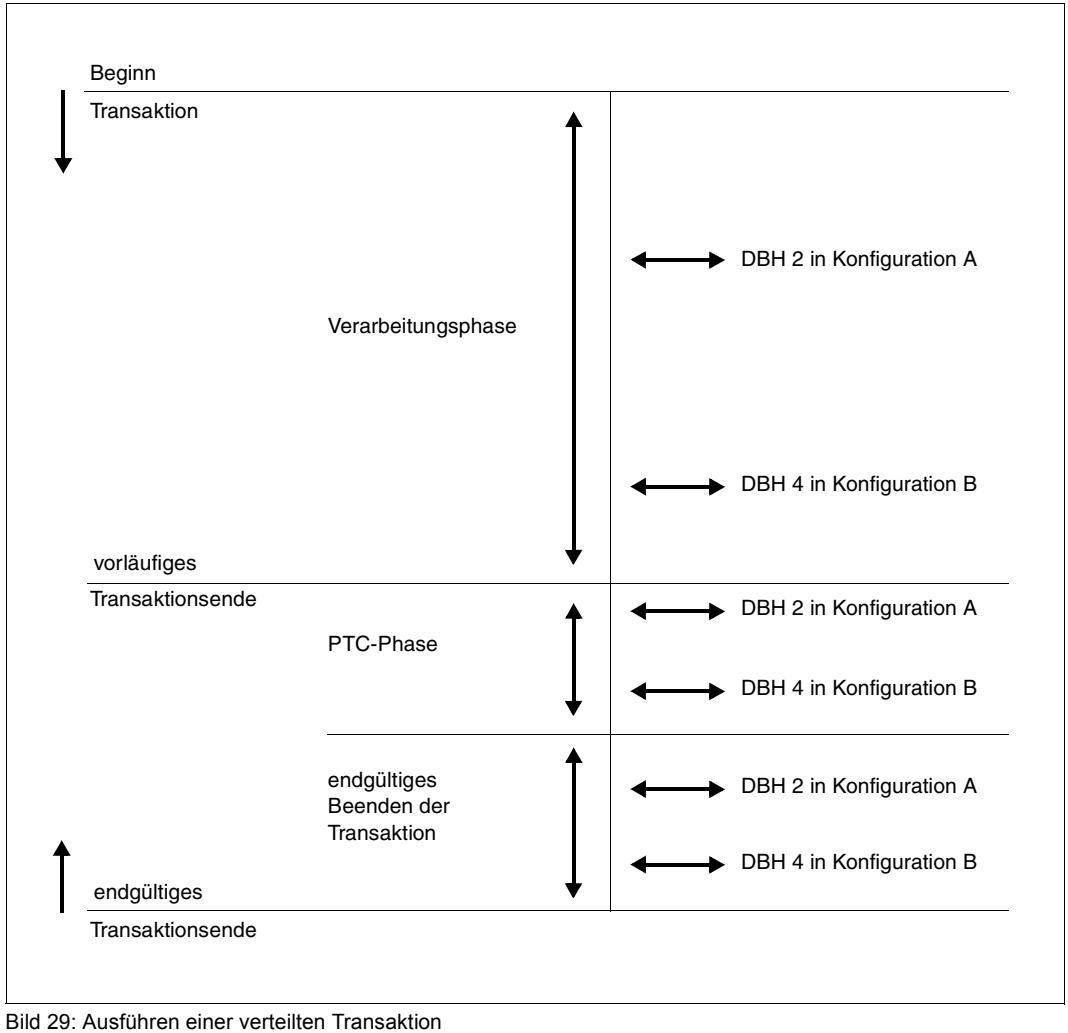


Bild 29: Ausführen einer verteilten Transaktion

„Konfiguration A“,
„Konfiguration B“ sowie
„DBH 2“ und
„DBH 4“ in Bild 29
beziehen sich auf die in Bild 28 dargestellte Situation.

Verarbeitungsphase
Die erste transaktionseinleitende SQL-Anweisung bzw. die Anweisung BTA bei CALL-DML und die übrigen SQL- bzw. CALL-DML-Anweisungen der Transaktion werden ausgeführt.

PTC-Phase (Prepare To Commit)
: Die Verarbeitungsphase innerhalb der Transaktion ist abgeschlossen. Die durchgeführten Änderungen werden ausfallsicher protokolliert, sind jedoch noch reversibel. Erst wenn „Prepared To Commit“ von allen beteiligten DBHs bestätigt worden ist, kann das endgültige Transaktionsende eingeleitet werden.

Phase „Endgültiges Beenden der Transaktion“
: Die Transaktion wird irreversibel abgeschlossen: Alle Änderungen werden festgeschrieben oder zurückgesetzt.

Master-DCN und Sicherungsdatei

Master-DCN

Nach einem Rechnerausfall ist zur Erhaltung der Transaktionskonsistenz ein Wiederanlauf von SESDCN (siehe „SESDCN-Wiederanlauf“ auf Seite 292) erforderlich. Koordination und Durchführung des SESDCN-Wiederanlaufs sind Aufgaben von SESDCN. Dabei überwacht SESDCN auch das Anlegen von Sicherungsdateien. Wenn SESDCN in der Konfiguration mehrfach geladen wurde, übernimmt der zuerst geladene SESDCN diese Aufgabe. Der SESDCN, der den Wiederanlauf durchführt, heißt Master-DCN.

Sicherungsdatei

Für den Wiederanlauf nach Systemausfall muss eine Sicherungsdatei vorhanden sein, die die Wiederanlaufdaten enthält.

Der Master-DCN prüft, ob die Sicherungsdatei bereits vorhanden ist.

- Falls noch keine entsprechende Datei vorhanden ist, wird die Sicherungsdatei beim erstmaligen Starten des Master-DCN unter dem über den Linknamen SESDLG zugewiesenen Dateinamen oder, falls keine Zuweisung über diesen Linknamen vorliegt, unter dem Standarddateinamen in der Kennung angelegt, in der der Master-DCN geladen ist.
  Dabei kann sie über die SET-DCN-OPTIONS-Anweisung SESDLG-PASSWORD mit einem Kennwort versehen werden. Der Kennwortschutz gilt für lesende und schreibende Zugriffe.
- Falls die Sicherungsdatei bereits vorhanden ist und noch nicht über einen Kennwortschutz verfügt, kann ein Kennwortschutz nur mit dem BS2000-Kommando MODIFY-FILE-ATTRIBUTES erreicht werden. Der Kennwortschutz sollte dann wie beim Parameter SESDLG-PASSWORD für Lesen und Schreiben gelten.
  Beim Öffnen einer bereits vorhandenen Sicherungsdatei stellt der Master-DCN fest, ob ein Wiederanlauf notwendig ist.

Im Folgenden sind Namenskonventionen, Zugriffsmethode und Standardzuweisung für die Sicherungsdatei von SESDCN angegeben:

Standardname	SES.DLG*k*
Linkname	SESDLG
Zugriffsmethode	PAM-shared (bzw. SHAREUPD)
Primär-/Sekundärzuweisung	9 / 24

Die Sicherungsdatei für SESDCN führt als Namensbestandteil den Konfigurationsnamen *k*, der sie der Konfiguration zuordnet, zu der der Master-DCN gehört.

Der Systemverwalter kann die Sicherungsdatei auch mit dem SDF-Kommando CREATE-FILE unter einem beliebigen Dateinamen anlegen und mit dem Linknamen SESDLG zuweisen.

SESDCN-Wiederanlauf

In der Sicherungsdatei ist vermerkt, ob die letzte SESDCN-Session ordnungsgemäß beendet wurde. Beim darauffolgenden Start von SESDCN öffnet der Master-DCN die Sicherungsdatei und prüft, ob ein Wiederanlauf durchgeführt werden muss. Der SESDCN-Wiederanlauf setzt voraus, dass die an der verteilten Verarbeitung beteiligten DBHs wiederanlauffähig sind.
Da ein SESDCN-Wiederanlauf die logische Fortsetzung des vorangegangenen SESDCN-Laufs ist, gelten alle Ladeoptionen des vorangegangenen Laufs.

Behandlung verteilter Transaktionen

Wie SESDCN beim Wiederanlauf mit verteilten Transaktionen verfährt, hängt ab von der Bearbeitungsphase, in der der vorangegangene SESDCN-Lauf abgebrochen wurde, und vom Ergebnis des dem SESDCN-Wiederanlauf vorangegangenen DBH-Wiederanlaufs.

Je nach Zeitpunkt des Systemausfalls behandelt der DBH beim Wiederanlauf die Transaktionen unterschiedlich:

- Der DBH setzt eine Transaktion beim Wiederanlauf zurück, wenn sich die Transaktion in der Verarbeitungsphase befand und die PTC-Phase aus Sicht des DBH noch nicht abgeschlossen war.
- Transaktionen, die sich beim Systemausfall in einer abgeschlossenen PTC-Phase befanden, bleiben beim DBH-Wiederanlauf in diesem Zustand. Erst der SESDCN-Wiederanlauf entscheidet auf Grund der Information in der Sicherungsdatei, ob die Transaktion zurückgesetzt oder beendet werden soll.

Wiederanlauf von SESDCN auf einem fremden Rechner

Der Wiederanlauf von SESDCN kann auf dem Rechner erfolgen, auf dem SESDCN ursprünglich gestartet wurde (Kaltstartrechner) oder auch auf einem anderen Rechner. Voraussetzung für den Wiederanlauf auf einem anderen Rechner ist die Vergabe netzweit eindeutiger Konfigurationsnamen bei der Systemgenerierung.

Für den Wiederanlauf auf einem anderen Rechner ist der Umzug der gesamten Konfiguration auf diesen Rechner erforderlich. Das bedeutet, dass alle zur Konfiguration gehörenden Anwenderprogramme, SESDCNs, DBHs, die Sicherungsdatei sowie alle DBH-spezifischen Dateien auf diesen Rechner gebracht werden müssen. Alle benötigten Tasks auf dem neuen Rechner müssen wiederanlauffähig sein. Außerdem muss die Systemumgebung für den DBH auf dem neuen Rechner die gleiche sein, d.h. die DBH-Kennung muss auf dem alten und neuen Rechner übereinstimmen. Ebenso müssen die DB-Kennungen auf dem alten und neuen Rechner übereinstimmen.

Aktualisierung der Verteilregeln

Erfolgt der SESDCN-Wiederanlauf auf einem anderen Rechner, so ändert sich aus Sicht der anderen Konfigurationen der Standort der Wiederanlauf-Konfiguration im Rechnernetz. Der Master-DCN der Wiederanlauf-Konfiguration ersetzt deshalb automatisch in den Verteilregeln der ihm bekannten und erreichbaren Remote-Rechner den alten Rechnernamen durch den Namen des Wiederanlaufrechners. Somit ist die Kommunikation mit den auf diesen Rechnern liegenden Konfigurationen weiterhin möglich.

Sind zum Zeitpunkt des SESDCN-Wiederanlaufs nicht alle Remote-Rechner im Netz für den Master-DCN erreichbar, so kann der Master-DCN nicht alle Verteilregeln automatisch ändern. Deshalb muss der Systemverwalter in diesem Fall einige Verteilregeln per Administrationsanweisung aktualisieren (siehe Handbuch „Datenbankbetrieb“).

Führen nach einem Rechnerausfall mehrere Konfigurationen ihren SESDCN-Wiederanlauf auf einem anderen Rechner durch, so ist eine vollständige Aktualisierung der Verteilregeln durch die zugehörigen Master-DCNs ebenfalls nicht möglich. Auch in diesem Fall muss der Systemverwalter einige Verteilregeln mit SESDCN-Administrationsanweisungen aktualisieren.

8.4.5 Versionsübergreifende verteilte Verarbeitung

Die SESAM/SQL-Versionen ab V4.0 können unbeschränkt und mit einer maximalen Nachrichtenlänge von 64000 Byte miteinander kommunizieren (siehe auch Seite 278).

Bei versionsübergreifender, verteilter Verarbeitung müssen Anwendung und DBH grundsätzlich in verschiedenen Konfigurationen ablaufen. Diese können sich auf demselben Rechner oder auf verschiedenen Rechnern befinden.

Zugriffe von Anwenderprogrammen mit CCSN=*NONE bzw. von Anwenderprogrammen aus SESAM/SQL < V5.0 auf eine Datenbank mit CCSN werden mit SQLSTATE abgewiesen. Eine versionsübergreifende, verteilte Verarbeitung ist nur für Datenbanken ohne CCSN möglich (CODE_TABLE hat den Wert _NONE_).

8.5 Dateibehandlung durch den DBH

Die Arbeit mit dem DBH erfolgt über Dateien, die jeweils eine spezifische Aufgabe übernehmen. Mit der Konfigurationsdatei können Sie dem DBH Startparameter zuweisen. CATID-Liste und MAIL-Parameterdatei enthalten Steuerinformationen für den DBH. Während einer Session legt der DBH selbst Dateien an, in denen er Informationen über seine Arbeit speichert. Diese Dateien dienen unter anderem dazu, Arbeitsschritte abzusichern, Aktionen durch den Anwender zurückzuverfolgen und auf den DBH im laufenden Betrieb Einfluss zu nehmen.

8.5.1 Angabe einer CATID-Liste

Mit SESAM/SQL-Server ist es möglich, die interne Suche nach Dateien auf bestimmte CATIDs zu beschränken. Dazu stellen Sie die gewünschten CATIDs in einer Datei zu einer Liste zusammen. Diese Datei können Sie dem DBH mit dem Kommando ADD-FILE-LINK und dem Linknamen SESAMCID zuweisen.

Die CATID-Liste wird beim ersten Zugriff auf eine Datei ausgewertet. Änderungen in der Datei werden wirksam, wenn der DBH neu gestartet oder die Administrationsanweisung MODIFY-CATID-LIST eingegeben wird. Die CATID-Liste wird mit der Administrationsanweisung SHOW-CATID-LIST ausgegeben (siehe jeweils Handbuch „Datenbankbetrieb“).

Zum Erstellen der CATID-Liste muss eine SAM-Datei mit variabler Satzlänge verwendet werden. Jeder Satz darf nur eine 1 bis 4 Zeichen lange CATID enthalten. Der Zeichenvorrat ist auf A ... Z, 0 ... 9 beschränkt, ohne Doppelpunkte und Leerzeichen. Es werden maximal 50 CATIDs ausgewertet. Die Angabe von wildcards ist nicht möglich.

Der Inhalt der CATID-Datei wird von SESAM/SQL nicht auf Richtigkeit geprüft. Enthält die Datei syntaktisch falsche oder im BS2000 nicht bekannte CATIDs, wird eine Fehlermeldung auf SYSLST ausgegeben. Ist die CATID-Datei leer, wird ausschließlich das Default-Pubset verwendet.

Die CATID des Standard-Pubset der Kennung muss grundsätzlich in die Liste aufgenommen werden. Gibt der Anwender die CATID des Standard-Pubset nicht an, wird sie von SESAM/SQL automatisch aufgenommen.

Wenn eine Datei neu angelegt werden soll, prüft SESAM/SQL nach folgendem Muster, ob diese Datei schon vorhanden ist:

- SESAM/SQL sucht die Datei auf allen CATIDs der CATID-Liste. Wird die Datei auf einer der CATIDs gefunden, wird sie nicht neu angelegt, sondern so verwendet.
- Ist die CATID der neu anzulegenden Datei nicht in der CATID-Liste, wird eine Fehlermeldung ausgegeben.

- Ist die CATID der neu anzulegenden Datei in der CATID-Liste und die Datei auf keiner CATID der CATID-Liste vorhanden, wird sie neu angelegt.

Kann die gesuchte Datei nicht eindeutig zugeordnet werden, wird eine entsprechende Meldung ausgegeben.

Dem Utility-Monitor sollten Sie die gleiche CATID-Liste zuweisen wie dem DBH, damit in jedem Fall auf die gleiche Datenmenge zugegriffen wird.
Sie ordnen die CATID-Liste dem Utility-Monitor zu mit dem Kommando ADD-FILE-LINK und dem Linknamen SESAMCID. Die CATID-Liste wird beim ersten Zugriff auf eine Datei ausgewertet.
Änderungen in der CATID-Liste werden nur wirksam, wenn der Utility-Monitor neu gestartet wird. Mit der Administrationsanweisung MODIFY-CATID-LIST ändern Sie nur die CATID-Liste des DBH. Dieses Kommando hat keine Auswirkungen auf die CATID-Liste, die dem Utility-Monitor zugewiesen wurde.

8.5.2 Konfigurationsdatei

Eine Konfigurationsdatei ist eine BS2000-Datei, die der Anwender anlegt. In der Konfigurationsdatei des DBH kann er die Startparameter für den DBH eintragen. Auch für SESDCN, den Utility-Monitor und DBCON, das ist das Konnektionsmodul des Anwenderprogramms, können die Startparameter über eine Konfigurationsdatei zugewiesen werden.

SESAM/SQL unterscheidet mehrere Typen von Konfigurationsdateien:

1. eine Konfigurationsdatei für DBH-Startparameter
2. eine Konfigurationsdatei für SESDCN-Startparameter
3. eine Konfigurationsdatei für Startparameter des DBCON und des Utility-Monitors

Die Konfigurationsparameter aller oder mehrerer Komponenten können in einer globalen Konfigurationsdatei zusammengefasst werden.

In allen Fällen handelt es sich um eine SAM-Datei mit frei wählbarem Namen, in die der Anwender die gewünschten Parameter einträgt.

Kommentarzeilen werden bei 1. und 2. mit „//REMARK“, bei 3. mit „REMARK“ eingeleitet. Sie können aus einer beliebigen alphanumerischen Zeichenfolge bestehen. Ende des Kommentars ist das Zeilenende.
Die Kommentarzeilen dürfen nicht zwischen Fortsetzungszeilen stehen, da sie sonst als Teil der Anweisung interpretiert werden.

Vor dem Start des DBH weist der Anwender eine der folgenden Dateien zu:

- entweder die Konfigurationsdatei für DBH-Startparameter
  mit dem Linknamen SESCONF oder über die Systemdatei SYSDTA
- oder eine globale Konfigurationsdatei
  mit dem Kommando CONNECT-SESAM-CONFIGURATION.

Für SESDCN, DBCON und den Utility-Monitor wird in gleicher Weise verfahren.

Im Laufe der Startprozedur werden die eingetragenen Parameter an den DBH bzw. an DBCON oder an den Utility-Monitor übergeben.

Enthält ein auszuwertender Parameter der Konfigurationsdatei Syntaxfehler, so wird die Datei geschlossen. Die jeweils gestartete Komponente gibt eine entsprechende Fehlermeldung aus.

8.5.2.1 Konfigurationsdatei für DBH-Startparameter

Dieser Typ einer Konfigurationsdatei dient als Eingabedatei für Startparameter des DBH und enthält ausschließlich DBH-Startanweisungen und -Optionen. Sie legen für die aktuelle DBH-Session Betriebsmittel, Grenzwerte und Arbeitsregeln fest.

Syntax und Bedeutung der einzelnen Startparameter des DBH sind im Abschnitt „Steuerung und Überwachung der Session“ auf Seite 323 beschrieben.
Hinweise für den Eintrag in die Konfigurationsdatei sowie eine detaillierte Beschreibung der DBH-Startparameter finden Sie im Handbuch „Datenbankbetrieb“.

Sie können eine DBH-Konfigurationsdatei mit allen DBH-Optionen und ihren aktuellen Werten in der laufenden DBH-Session bequem mit der Administrationsanweisung SAVE-DBH-OPTIONS erstellen, siehe Handbuch „Datenbankbetrieb“.

8.5.2.2 Globale Konfigurationsdatei

Dieser Typ der Konfigurationsdatei fasst die Konfigurationsparameter mehrerer Komponenten in einer Datei zusammen, z.B. alle DBHs, SESDCNs und DBCONs einer Konfiguration. Bei der Installation von SESAM/SQL muss dafür die Voraussetzung geschaffen worden sein (siehe Handbuch „Datenbankbetrieb“).

Die Parameter sind in dieser Datei komponentenweise zu Parameterblöcken geordnet. Jeder Parameterblock wird durch einen einzeiligen Eintrag
CONFIGURATION-LINK=*komponente* eingeleitet.
komponente identifiziert die Komponente, zu der dieser Parameterblock gehört.
Diese Datei weisen Sie mit dem Kommando CONNECT-SESAM-CONFIGURATION der Komponente zu, für die die Datei Parameter enthält (siehe Handbuch „Datenbankbetrieb“).

Für die Anwendung der verschiedenen Parameter in der globalen Konfigurationsdatei gelten die gleichen Regeln wie für die Konfigurationsdateien einzelner SESAM/SQL-Komponenten, wie z.B. des Utility-Monitors oder des SESDCN.

Die Besonderheiten bei der Eingabe von Startparametern für SESDCN, DBCON und Utility-Monitor sind ab Seite 299 beschrieben.

Beispiel für eine globale Konfigurationsdatei

```
//REMARK LOAD OPTIONS FOR THE DBH *********************************
CONFIGURATION-LINK=SESDBB1
//SET-DBH-OPTIONS-
//    DBH-IDENTIFICATION=*PARAMETERS(-
//        CONFIGURATION-NAME=Z-
//       ,DBH-NAME=X-
//    )-
//   ,ADMINISTRATION=*PARAMETERS(-
//        ACCOUNTING=*PARAMETERS(-
.
.
.
//REMARK K_2_DBH_NEW_1 *******************************************
//ADD-SQL-DATABASE-CATALOG-LIST-
//    ENTRY-1=*CATALOG(-
//        CATALOG-NAME=AUFTRAGKUNDEN-
//       ,USER-ID=KENN1-
//       )-
.
.
.
//REMARK LOAD OPTIONS FOR USER PROGRAMS **********************
CONFIGURATION-LINK=SESDBB2
REMARK CONFIGURATION DATA ***************************************
CNF=Z
NAM=X
SQLOPT-PREFERRED-JOIN-METHOD=SORT-MERGE
PUF=64000
TOTAL-USERS=00128
REMARK A DB PROCESS IS ASSIGNED EXACTLY ONE UTM PROCESS ******
UTMVG=JA
```

Startparameter für SESDCN

Sie legen für die aktuelle SESDCN-Session die Verteilregel und „Zugriffspfade“ fest.

Syntax und Bedeutung der einzelnen Startparameter des SESDCN sind im Abschnitt „SESDCN-Steueranweisungen und Optionen“ auf Seite 327 beschrieben. Hinweise für den Eintrag in die Konfigurationsdatei sowie eine detaillierte Beschreibung der SESDCN-Startparameter finden Sie im Handbuch „Datenbankbetrieb“.

Sie können als Eingabedatei für Startparameter des SESDCN neben einer globalen Konfigurationsdatei auch eine eigene Konfigurationsdatei verwenden. Dieser Typ einer Konfigurationsdatei enthält dann ausschließlich SESDCN-Steueranweisungen und -Optionen.

Startparameter für DBCON- und Utility-Monitor

Sie können die Startparameter für das Konnektionsmodul (DBCON) einer ESQL- oder CALL-DML-Anwendung und für den Utility-Monitor über eine globale Konfigurationsdatei oder über eine eigene Konfigurationsdatei zuweisen. Dieser Typ einer Konfigurationsdatei kann Parameter für den Utility-Monitor oder DBCON oder beide Komponenten enthalten.

Beim Eintrag der Parameter in eine Konfigurationsdatei dieses Typs ist Folgendes zu beachten:

- in einer Zeile, bzw. in einem Satz, darf nur ein Parameter stehen
- der Parameter muss in Spalte 1 beginnen und darf keine Leerzeichen enthalten
- jeder Parameter darf nur einmal in die Datei eingetragen werden
- die Reihenfolge der Parameter ist beliebig

Sowohl DBCON als auch der Utility-Monitor werten jeweils nur ihre spezifischen Parameter aus. Als Unterscheidungskriterium beginnen die Parameter des Utility-Monitors mit dem Präfix „SEE“, die Startparameter des DBCON enthalten kein Präfix. Der Utility-Monitor liest zusätzlich zu den mit „SEE“ gekennzeichneten Parametern den DBH-Namen und den Konfigurationsnamen (Konnektionsmodul-Parameter NAM bzw. CNF).

DBCON-Parameter

Wichtige DBCON-Parameter (Konnektionsmodul-Parameter) sind der DBH-Name und der Konfigurationsname. Diese beiden Parameter ordnen die Anwendung, zu welcher das Konnektionsmodul gebunden ist, einem DBH und einer Konfiguration zu (siehe Abschnitt „Konnektionsmodul-Parameter“ auf Seite 274).

Eine Konfigurationsdatei mit Konnektionsmodul-Parametern gibt es nur für solche Anwendungen, die mit dem independent DBH zusammenarbeiten, nicht aber für linked-in Anwendungen. Bei linked-in Anwendungen sind die wichtigsten Parameter, DBH-Name und Konfigurationsname, als DBH-Optionen in der Konfigurationsdatei des linked-in DBH enthalten.

Im TIAM- und im DCAM-Betrieb muss der Anwender die Konnektionsmodul-Parameter vor dem Start der Anwendung über die Konfigurationsdatei zuweisen, es sei denn, er wünscht die voreingestellten Standardwerte als Parameter. Der Anwender von CALL-DML-Anwendungen kann einige Konnektionsmodul-Parameter auch über spezielle CALL-DML-Aufrufe übergeben (siehe Handbuch „CALL-DML Anwendungen“). Der Einsatz der Konfigurationsdatei bietet aber den Vorteil der Flexibilität, denn ein Anwenderprogramm kann ohne Eingriff in die Source und ohne erneuten Binderlauf einem anderen DBH oder einer anderen Konfiguration zugeordnet werden. Außerdem kann er auch Konfigurationsparameter mehrerer Komponenten in einer globalen Konfigurationsdatei zusammenfassen.

Dieselbe Konfigurationsdatei kann sowohl TIAM- , UTM- und DCAM-Anwendungen zugewiesen werden. Enthält die Konfigurationsdatei für eine TIAM-Anwendung DCAM- oder UTM-spezifische Konnektionsmodul-Parameter, so werden diese ignoriert. Entsprechendes gilt für DCAM- und UTM-Anwendungen.

Eine Auflistung der Syntax und Bedeutung aller Konnektionsmodul-Parameter und SQL-Hinweise für TIAM- und DCAM-Anwendungen finden Sie ab Seite 275.

Utility-Monitor-Parameter

Utility-Monitor-Parameter, die sog. Konfigurationsdaten, steuern den Programmablauf des Utility-Monitors.

Als Pflichtparameter muss der Anwender den Berechtigungsschlüssel für das Arbeiten mit dem Utility-Monitor bzw. den Berechtigungsschlüssel für die Administration über die CALL-Schnittstelle angeben. Außerdem gibt es verschiedene Wahlparameter.

Weist der Anwender die Konfigurationsdatei nicht zu, so muss er die Konfigurationsdaten, zumindest die Berechtigung für das Arbeiten mit dem Utility-Monitor, im laufenden Betrieb des Monitors bekannt geben.

Sie sollten den DBH-Namen und den Konfigurationsnamen (Konnektionsmodul-Parameter NAM bzw. CNF) in die Konfigurationsdatei eintragen, da diese Angaben in der Maske CNF - CONFIGURATION zwar angezeigt werden, aber nicht geändert werden können. Wenn der DBH-Name und der Konfigurationsname nicht in der Konfigurationsdatei angegeben sind, wird der Utility-Monitor mit dem Standardnamen des DBH und mit dem Standardwert des Konfigurationsnamens gestartet (siehe Handbuch „Datenbankbetrieb“).

Nachfolgend sind die wichtigsten Konfigurationsdaten des Utility-Monitor aufgelistet. Eine vollständige Beschreibung der Konfigurationsdaten finden Sie im Handbuch „Utility-Monitor“.

SEE-AUTHID=authorization
: Berechtigungsschlüssel für das Arbeiten mit dem Utility-Monitor.
Pflichtparameter

SEE-INST-LOGGING={ON | OFF}
: Protokollierung in Anweisungsdatei ein- bzw. ausschalten.
Standardwert: OFF

SEE-EXECUTE={ON | OFF}
: Ausführung der protokollierten Anweisungen veranlassen bzw. verhindern.
Standardwert: ON

SEE-ERROR={ON | OFF}
: Reaktion auf DBH-Fehlermeldung beim Abarbeiten der Anweisungsdatei festlegen (Abbruch oder Weiterverarbeitung).
Standardwert: ON

SEE-COPY={ON | OFF}
: Automatische Sicherung von Datenbankobjekten ein- bzw. ausschalten.
Standardwert: ON

SEE-ADMIN=password
: Kennwort für die Administration über das Administrationsprogramm SESADM.
Pflichtparameter für die Administration über die CALL-DML-Schnittstelle.

8.5.3 MAIL-Parameterdatei

Die MAIL-Parameterdatei ist eine BS2000-SAM-Datei, die der Anwender anlegt und beim Start dem DBH über den Linknamen SESMAIL zuweist, siehe Abschnitt „E-Mail-Versand wichtiger Informationen der DBH-Session“ auf Seite 335.

Detaillierte Informationen zum Aufbau der MAIL-Parameterdatei und zur E-Mail-Ausgabe des DBH finden Sie im Handbuch „Datenbankbetrieb“

8.5.4 DBH-spezifische Dateien

DBH-spezifische Dateien sind sessionbezogene Dateien, die der DBH selbst anlegt und die fest einem DBH zugeordnet sind.

Die Standardkennung für DBH-spezifische Dateien ist die Startkennung des DBH. Der Systemverwalter kann jedoch veranlassen, dass DBH-spezifische Dateien, abgesehen von den Cursor-Dateien, unter einer anderen Kennung angelegt werden. In dem Fall muss er die entsprechenden Dateien vor dem Starten des DBH mit dem jeweiligen Linknamen zuweisen.

Medienkatalog

Der Medienkatalog des DBH enthält Speicherinformationen für folgende DBH-spezifische Dateien:

- die Transaktionssicherungsdateien (2 TA-LOG-Dateien)
- die Wiederanlauf-Sicherungsdatei (WA-LOG-Datei)
- die Cursor-Dateien

Er beschreibt, auf welchem Datenträger, unter welcher Katalogkennung und mit welchen Speicherzuweisungen diese Dateien angelegt werden sollen.

Die Informationen im Medienkatalog werden nur ausgewertet, wenn eine Datei nicht bereits zu Sessionbeginn angelegt ist:
Die DBH-spezifischen Dateien werden nur dann auf dem spezifizierten Medium angelegt, wenn sie nicht bereits zu Sessionbeginn auf einem anderen Medium liegen.

Der Systemverwalter richtet den Medienkatalog über die DBH-Option MEDIA-CATALOG ein. Der Medienkatalog liegt nicht in Dateiform vor. Er ist nur für eine DBH-Session gültig und wird im Speicher des DBH gehalten.

8.5.4.1 Übersicht über DBH-spezifische Dateien

Die folgende Tabelle zeigt eine Übersicht über alle DBH-spezifischen Dateien. Die Platzhalter in den Dateinamen haben folgende Bedeutung:

ssss Sessionidentifikation, deren Standardwert der BS2000-Prozessfolgenummer (tsn) entspricht

iiii vierstelliger Startwert für Dateizähler

k Konfigurationsname (ein byte)

n DBH-Name (ein byte)

st Service-Task-Identifikation (zwei byte, von 01 bis 64); wird nur innerhalb einer Service-Task versorgt

Standarddateiname	Linkname	Primär-/ Sekundär-zuweisung	Zugriffs-methode	Bedeutung
SESAM.CO-LOG.*ssss*.*iiii*	----------	192/24	PAM/ BTAM	Datei für die Auftrags-protokollierung (independent DBH)
SESAM*kn*.CURSOR.0001 SESAM*kn*.CURSOR.0002	----------	9/24 bzw. lt. Medien-katalog	PAM	Arbeitsdatei für Zwischen- und Endergebnisse von Wiedergewinnungs-anweisungen (independent DBH)
SESLK*kn*.CURSOR.0001 SESLK*kn*.CURSOR.0002	----------	9/24 bzw. lt. Medien-katalog	PAM	Arbeitsdatei für Zwischen- und Endergebnisse von Wiedergewinnungs-anweisungen (linked-in DBH)
SESAM*kn*.TA-LOG1 SESAM*kn*.TA-LOG2	TALOG1 TALOG2	lt. Medien-katalog	PAM	Transaktionssicherungs-dateien (independent DBH)
SESLK*kn*.TA-LOG1 SESLK*kn*.TA-LOG2	TALOG1 TALOG2	lt. Medien-katalog	PAM	Transaktionssicherungs-dateien (linked-in DBH)
SESAM*kn*.WA-LOG	WALOG	lt. Medien-katalog	PAM	Wiederanlauf-Sicherungsdatei (independent DBH)
SESLK*kn*.WA-LOG	WALOG	lt. Medien-katalog	PAM	Wiederanlauf-Sicherungsdatei (linked-in DBH)

Tabelle 52: DBH-spezifische Dateien

Sollen die DBH-spezifischen Dateien nicht unter den im Medienkatalog angegebenen Speichermedien angelegt werden, kann der Systemverwalter diese Angaben umgehen, indem er vor dem Start des DBH die Dateien mit dem CREATE-FILE-Kommando einrichtet.

Bei TA-LOG-, WA-LOG- und Cursor-Dateien eines linked-in DBH in **einer** Service-Task wird das Präfix SESLK*kn* durch SESLK*knst* ersetzt.

8.5.4.2 Transaktionssicherungsdateien TA-LOG

Die Transaktionssicherungsdateien (TA-LOG-Dateien) sind DBH-spezifische Dateien, in die der SESAM/SQL-DBH Informationen für die Ausfallsicherung und das Zurücksetzen von Transaktionen schreibt (siehe Abschnitt „Transaktionssicherung“ auf Seite 205). Insbesondere schreibt der DBH solche Informationen in die TA-LOG-Dateien, die die physikalische und logische Konsistenz der Daten in den Datenbanken gewährleisten.

Dazu gehören:

- After-Images, das sind Blöcke bzw. Sätze oder Satzteile eines Blocks, die durch Datenänderungen entstanden sind
- logische Before-Images, also Satzteile innerhalb eines Datenbankblocks vor der ersten Änderung
- Konsistenzpunkte, die der DBH bei jeder Anweisung „Ende Transaktion“ setzt.

Der DBH legt für jede DBH-Session mit Transaktionssicherung 2 TA-LOG-Dateien an. Die TA-LOG-Dateien werden nur dann nicht angelegt, wenn der Systemverwalter per DBH-Option die Transaktionssicherung ausgeschaltet hat. Speicherinformationen für die TA-LOG-Dateien sind im Medienkatalog des DBH hinterlegt.

Die Standardnamen der TA-LOG-Dateien lauten folgendermaßen:

DBH-Variante	Standardname	Linkname
independent DBH	SESAM*kn*.TA-LOG1 SESAM*kn*.TA-LOG2	TALOG1 TALOG2
linked-in DBH	SESLK*kn*.TA-LOG1 SESLK*kn*.TA-LOG2	TALOG1 TALOG2

Tabelle 53: Standardname der TA-LOG-Datei

Bei Nutzung der Utility-Funktion RECOVER werden zusätzlich die Dateien SESLK*knst*.TA-LOG1 und SESLK*knst*.TA-LOG2 angelegt.

Weitere Informationen zu Namenskonventionen, Standardzuweisung und Zugriffsmethode bzgl. der TA-LOG-Dateien finden Sie in der Tabelle „DBH-spezifische Dateien“ auf Seite 303.

TA-LOG-Dateien sind Ein-/Ausgabe-intensiv, in Ausnahmefällen kann es zu Ein-/Ausgabe-Engpässen kommen. TA-LOG-Dateien sollten deshalb auf einem eigenen, möglichst schnellen Gerät liegen.

Die Größe der Schreibeinheit für die TA-LOG-Dateien ist abhängig vom Plattentyp. SESAM/SQL nutzt die maximal mögliche Ein-/Ausgabe-Länge (64 bis 160 KB). Information über die maximale Ein-/Ausgabe-Länge in Half-Pages (2KB) erhalten Sie mit dem BS2000-Kommando

```
/SHOW-PUBSET-CONFIGURATION PUBSET=<catid>,INFORMATION=*PUBSET-FEATURES.
```

Transaktionssicherungsdatei für SQL-DDL-Anweisungen DDL-TA-LOG

Für SQL-Anweisungen zur Schemadefinition und -verwaltung und zur Verwaltung der Speicherstrukturen existiert eine eigene Transaktionssicherungsdatei. Ihr Standardname ist: :*catid*:*userid*.*catalogname*.*spacename*.DDLTA.

Es handelt sich um eine Loggingdatei, die dem Space zugeordnet ist, auf den die SQL-Anweisung wirkt. Sie enthält die physikalischen Before Images der o.g. SQL-Anweisungen. Die Datei wird am Ende der Transaktion gelöscht, wenn sie nicht durch den Anwender angelegt worden war. Siehe auch Seite 206.

8.5.4.3 Wiederanlauf-Sicherungsdatei WA-LOG

Die Wiederanlauf-Sicherungsdatei (WA-LOG-Datei) ist eine zusätzliche DBH-spezifische Sicherungsdatei, die den Wiederanlauf steuert. Sie enthält:

- DBH-Optionen
- Informationen zu Konsistenzpunkten
- physikalische Before Images
- Informationen über den Fortschritt des Wiederanlaufs
- Information zur openUTM-Synchronisation.

Der Standardname der WA-LOG-Datei lautet folgendermaßen:

DBH-Variante	Standardname	Linkname
independent DBH	SESAM*kn*.WA-LOG	WALOG
linked-in DBH	SESLK*kn*.WA-LOG	WALOG

Tabelle 54: Standardname der WA-LOG-Datei

Bei Nutzung der Utility-Funktion RECOVER wird zusätzlich die Datei SESLK*knst*.WA-LOG angelegt.

Der DBH legt für jede DBH-Session mit Transaktionssicherung eine WA-LOG-Datei an. Die WA-LOG-Datei wird nur dann nicht angelegt, wenn der Systemverwalter per DBH-Option die Transaktionssicherung ausgeschaltet hat. Speicherinformationen für die WA-LOG-Datei sind im Medienkatalog des DBH hinterlegt.

Weitere Informationen zu Namenskonventionen, Standardzuweisung und Zugriffsmethode bzgl. der WA-LOG-Datei finden Sie in der Tabelle „DBH-spezifische Dateien“ auf Seite 303.

8.5.4.4 Cursor-Dateien für Wiedergewinnungsanweisungen

Eine Cursor-Datei ist eine DBH-spezifische Arbeitsdatei, die für die Bearbeitung von Wiedergewinnungsanweisungen erforderlich ist.

Cursor-Dateien haben bei SESAM/SQL zweierlei Bedeutung:

- einerseits speichert der SESAM/SQL-DBH die Zwischenergebnisse von Wiedergewinnungsanweisungen in Cursor-Dateien.
- andererseits kann auch ein Auftraggeber Ergebnisse von CALL-DML-Suchfragen in einer Cursor-Datei ablegen und später weiter einschränken. Solche CALL-DML-spezifischen Cursor-Dateien werden durch ein Dateikennzeichen identifiziert (siehe Handbuch „CALL-DML Anwendungen“).

Speicherinformationen für die internen Cursor-Dateien sind im Medienkatalog des DBH hinterlegt.

SESAM/SQL legt maximal zwei Cursor-Dateien an. Die Standardnamen der internen Cursor-Dateien lauten folgendermaßen:

DBH-Variante	Standardname
independent DBH	SESAM*kn*.CURSOR.0001
	SESAM*kn*.CURSOR.0002
linked-in DBH	SESLK*kn*.CURSOR.0001
	SESLK*kn*.CURSOR.0002

Tabelle 55: Standardnamen der internen Cursor-Dateien

In die Datei mit dem Suffix „0001“ werden die Endergebnisse von CALL- DML-Suchfragen und von SQL-Anweisungen, die sich auf einen Cursor beziehen, abgelegt. Ferner wird diese Datei bei Bedarf als interner temporärer Arbeitsbereich zur Ablage von Zwischenergebnissen verwendet.

In die Datei mit dem Suffix „0002“ werden die Zwischenergebnisse von Sekundärindex-Auswertungen bei CALL-DML- und SQL-Wiedergewinnungsanweisungen abgelegt.

Beide Dateien werden erst bei Bedarf angelegt.

Weitere Informationen zu Namenskonventionen, Standardzuweisung und Zugriffsmethode bzgl. der „internen“ Cursor-Dateien finden Sie in der Tabelle „DBH-spezifische Dateien“ auf Seite 303.

8.5.4.5 CO-LOG-Datei für die Auftragsprotokollierung

Die Auftragsprotokolldatei (CO-LOG-Datei) ist eine DBH-spezifische Logging-Datei, die für die Auftragsprotokollierung zur Verfügung steht (siehe Abschnitt „Auftragsprotokollierung auswerten mit SESCOSP“ auf Seite 336).

Der DBH eröffnet die CO-LOG-Datei, sobald der Systemverwalter die Auftragsprotokollierung per Administrationsanweisung einschaltet. Schaltet der Systemverwalter die Auftragsprotokollierung zwischenzeitlich ab und aktiviert sie in derselben Session wieder, so schließt der DBH die CO-LOG-Datei und eröffnet später eine weitere CO-LOG-Datei.

Der Standardname der CO-LOG-Datei lautet folgendermaßen:

DBH-Variante	Standardname
independent und linked-in DBH	SESAM.CO-LOG.*ssss*.*iiii*

Tabelle 56: Standardname der CO-LOG-Datei

Über die DBH-Option SESSION-LOGGING-ID kann der Systemverwalter sowohl für die Session-Identifikation (*ssss*) als auch für den Dateizähler (*iiii*) eigene Werte vergeben.

Weitere Informationen zu Namenskonventionen, Standardzuweisung und Zugriffsmethode bzgl. der CO-LOG-Datei, finden Sie in der Tabelle „DBH-spezifische Dateien“ auf Seite 303.

8.5.5 Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen

Mit SESAM/SQL können Sie die Datenbank in einer anderen Benutzerkennung als der DBH-Kennung, der sogenannten DB-Kennung, ablegen. Liegt der Catalog auf der DB-Kennung, dann wird bei Ausführung von einigen Utility- oder DDL-Anweisungen versucht, neu anzulegende Dateien und Jobvariablen ebenfalls in der DB-Kennung anzulegen.

Um dies zu ermöglichen, muss der Datenbankverwalter Vorbereitungen treffen.
Es gibt zwei Möglichkeiten:

- In der DB-Kennung die Miteigentümerschaft für die DBH-Kennung erklären
- Dateien und Jobvariablen mit BS2000-Kommandos anlegen

In der DB-Kennung die Miteigentümerschaft für die DBH-Kennung erklären

Diese Vorgehensweise wird empfohlen. Voraussetzung ist der Einsatz des Softwareproduktes SECOS.

Der Datenbankverwalter erklärt in der DB-Kennung die DBH-Kennung als Miteigentümer der betreffenden Objekte (Dateien und Jobvariablen). Die DBH-Kennung hat damit dieselben Rechte in Bezug auf die Objekte wie die DB-Kennung. Dies schließt insbesondere auch die Rechte ein, die Objekte anzulegen oder ein Kennwort dafür zu vergeben.

Beispiel

Die DBH-Kennung <dbh-id> soll das Recht haben, unter der DB-Kennung <db-id> Dateien für den Catalog <db-cat> anzulegen, zu verwalten und zu löschen.

Lösung

<db-id> definiert ein Bedingungsguard <db-cond>, das <dbh-id> zeitlich unbegrenzten Zugriff gewährt:

```
/create-guard <db-cond>,user-inf='Zugriffsbedingungen fuer DBH'
/add-access-conditions guard-name=<db-cond>, -
/                      subjects=*user(user-identification=<dbh-id>)
```

Dann definiert <db-id> im aktiven Regelbehälter SYS.UCF eine Miteigentümerschutzregel. Diese gibt an, dass die Zugriffsbedingungen für Dateien, deren Name dem Muster „<db-cat>*“ entspricht, im Bedingungsguard <db-cond> festgelegt sind.

```
/add-coowner-protection-rule rule-container-guard=sys.ucf, -
/            protection-rule=rule1, -
/            protect-object=*parameters(name=<db-cat>*,-
/                                       condition-guard=<db-cond>)
```

Für Jobvariablen (z.B. SESAM.*replikat*.NEXT-REPL-LOG) kann in derselben Weise die Miteigentümerschaft definiert werden. Der aktive Regelbehälter für Jobvariablen hat den Namen SYS.UCJ.

Die Miteigentümerschaft muss für jede BS2000-Katalogkennung (cat-id) separat definiert werden.

Nähere Informationen zur SECOS-Funktion „Miteigentümerschutz (Co-owner protection)“ finden Sie im SECOS-Handbuch „Security Control System - Zugangs- und Zugriffskontrolle“, Kapitel „Guards-Schutz“.

Dateien und Jobvariablen mit BS2000-Kommandos anlegen

Der Datenbankverwalter legt mit dem BS2000-Kommando CREATE-FILE auf der DB-Kennung einen Katalogeintrag für jede betreffende Datei an.
Jobvariablen werden mit dem BS2000-Kommando CREATE-JV auf der DB-Kennung angelegt.

Die Dateien und Jobvariablen müssen mehrbenutzbar (Operand USER-ACCESS=*ALL-USERS) und mit dem Schreibrecht (Operand ACCESS=*WRITE) versehen sein.

Dateien und Jobvariablen mit BS2000-Kennwort

Bei definierter Miteigentümerschaft legt SESAM/SQL die Dateien mit dem angegebenen BS2000-Kennwort an.

Wenn der Datenbankverwalter die Dateien und Jobvariablen anlegt, so muss er das gewünschte BS2000-Kennwort vergeben (z.B. beim Einrichten der Dateien und Jobvariablen).

Betroffene Anweisungen und Dateien

Folgende Anweisungen können Dateien in der DB-Kennung anlegen oder löschen:

Anweisung / Prozess	**Dateien**
SQL-Anweisungen	
CREATE SPACE	Anwender-Space
DROP SPACE	Anwender-Space löschen
ALTER TABLE	Fehlerdatei

Tabelle 57: Dateien, die in der DB-Kennung angelegt werden können (Teil 1 von 2)

Utility-Funktionen	
CHECK FORMAL	Fehlerdatei
COPY (auf Platte) COPY CATALOG[_SPACE] COPY CATALOG[_SPACE] LOG	Sicherungsdateien von Catalog-Space und Anwender-Spaces CAT-REC-Kopie, CAT-LOG-Datei, DA-LOG Datei CAT-REC-Datei und CAT-REC-Kopie, falls diese zum ersten Mal angelegt werden (Catalog vorher ohne Logging)
CREATE CATALOG	Catalog-Space, CAT-LOG-Datei, (CAT-REC-Datei)
CREATE INDEX	Arbeitsdatei bei parallelem Indexaufbau
CREATE REPLICATION	Catalog-Space, Anwender-Spaces, CAT-REC-Datei des Replikats
EXPORT TABLE	Export-Datei
IMPORT TABLE	Arbeitsdatei bei parallelem Indexaufbau
LOAD	Fehlerdatei
RECOVER CATALOG RECOVER CATALOG_SPACE RECOVER SPACE	Catalog-Space, Anwender-Space, CAT-LOG-Datei, DA-LOG-Datei Catalog-Space, CAT-LOG Datei Anwender-Space, DA-LOG Datei
REFRESH SPACE	Anwender-Space
REORG SPACE REORG CATALOG_SPACE	Arbeitsdatei
UNLOAD	Ausgabedatei, Fehlerdatei
Administrationsanweisungen von SESADM bzw. Administrationskommando	
CHANGE-CATLOG bzw. CAW	CAT-LOG-Datei, DA-LOG-Datei, CAT-REC-Kopie
CHANGE-DALOG bzw. DAW	DA-LOG-Datei
DML-Anweisungen	
Erster ändernder Zugriff	DA-LOG-Datei

Tabelle 57: Dateien, die in der DB-Kennung angelegt werden können (Teil 2 von 2)

Nähere Hinweise finden Sie bei der Beschreibung der oben genannten Anweisungen.
Die DDL-TA-LOG-Datei wird immer in der DBH-Kennung angelegt.
SESAM/SQL löscht nur Dateien, die SESAM/SQL selbst angelegt hat. Dateien, die durch einen Anwender angelegt wurden, werden nicht gelöscht.

8.6 Puffer und Container des Data Base Handlers

Die differenzierte Puffer- und Container-Technik des Data Base Handlers (DBH) bietet eine wichtige Funktion für die Durchsatzsteigerung. Puffer und Container sind Hauptspeicherbereiche zur Zwischenspeicherung bestimmter Daten.

Die wichtigsten Puffer und Container des SESAM/SQL-DBH sind folgende:

- User-Data-Buffer - für Anwenderdaten
- System-Data-Buffer - für Systemzugriffsdaten
- Sicherungspuffer - für Transaktionssicherung und logische Datensicherung
- Cursor-Puffer - für Zwischenergebnisse von Wiedergewinnungsanweisungen
- Planpuffer - für SQL-Zugriffspläne
- Transfer-Container - für Frage-/Antwortbereiche von SQL-Scans und logischen Dateien
- Work-Container - für Arbeitsleisten

Ziel der Puffer- und Container-Technik ist es, Platten-Ein-/Ausgaben zu minimieren: Schreibaufträge, die sonst blockweise ausgeführt werden müssten, können gesammelt und zusammenhängend auf die Platte geschrieben werden.

Da Hauptspeicher nur in begrenztem Umfang vorhanden ist, können nicht alle zwischengespeicherten Datenblöcke resident gehalten werden. Die Datenblöcke der Pufferbereiche und des Work-Containers werden daher bei Pufferüberlauf verdrängt. Welche Blöcke verdrängt werden, entscheidet die Pufferverwaltung des DBH auf Grund von Kriterien, die den bestmöglichen Durchsatz zum Ziel haben. So legt der DBH für Daten mit unterschiedlichem Verdrängungsverhalten getrennte Pufferbereiche an, nimmt aber auch innerhalb der einzelnen Pufferbereiche die Verdrängung der Blöcke differenziert vor. Er vermeidet so, dass Blöcke mit häufig benötigten Informationen durch solche mit weniger häufig benötigten Informationen verdrängt werden.

Die folgenden Abschnitte beschreiben, welche Daten in den einzelnen Puffern und Containern zwischengespeichert werden.

8.6.1 User-Data-Buffer und System-Data-Buffer

Der SESAM/SQL-DBH verwaltet einen User-Data-Buffer, der für Anwenderdaten zur Verfügung steht und einen System-Data-Buffer für Systemzugriffsdaten. Verwaltungs- und Zugriffseinheit ist jeweils ein PAM-Block mit einer Größe von 4 Kbyte.

Die Blöcke des User-Data-Buffer sind aus Sicht des DBH untereinander gleichbedeutend, die Blöcke des System-Data-Buffer sind hierarchisch strukturiert. Deshalb verwaltet der DBH die Blöcke des System-Data-Buffer in unterschiedlichen Gewichtsklassen entsprechend ihrer Bedeutung für die Zugriffsoptimierung des DBH.

Die jeweilige Puffergröße kann der Systemverwalter beim Starten des DBH mit den entsprechenden DBH-Optionen bedarfsgerecht dimensionieren.

Verdrängung aus dem Hauptspeicher

Fehlt im Puffer Platz für die Speicherung neuer Daten, so werden im Standardfall Blöcke nach dem LRU-Prinzip (Least Recently Used-Prinzip) verdrängt. Dabei werden stets die ältesten Blöcke überschrieben, also die, auf die am längsten nicht mehr zugegriffen wurde. Auf diese Weise werden Blöcke, die ständig benötigt werden, gegenüber anderen Blöcken bevorzugt resident gehalten.

8.6.2 Sicherungspuffer

TA-LOG-Puffer

Für den Betrieb mit Transaktionssicherung legt der DBH den sogenannten TA-LOG-Puffer an. Der TA-LOG-Puffer ist als Wechselpuffer konzipiert, d.h., er besteht aus zwei Puffern, in die der DBH abwechselnd Sicherungsinformation schreibt. Sobald ein Puffer voll ist, wird sein Inhalt auf die aktuelle TA-LOG-Datei (siehe Abschnitt „Transaktionssicherungsdateien TA-LOG“ auf Seite 304) übertragen. Während des Übertragens schreibt der DBH die Sicherungsinformationen in den anderen Puffer.

CAT-LOG- und DA-LOG-Puffer

Der DBH führt je Datenbank einen sogenannten CAT-LOG- und DA-LOG-Puffer, in dem er die Anweisungen für das CAT- bzw. DA-Logging zwischenspeichert, bevor sie auf die CAT-LOG- bzw. DA-LOG-Datei übertragen werden (siehe Seite 218 und Seite 220).

8.6.3 Cursor-Puffer

Im Cursor-Puffer speichert der DBH Zwischenergebnisse von Wiedergewinnungsanweisungen.

Ist der Cursor-Puffer zu klein dimensioniert, so werden bei Pufferüberlauf die Daten des Cursor-Puffers in eine oder mehrere interne Cursor-Dateien (siehe Abschnitt „Cursor-Dateien für Wiedergewinnungsanweisungen“ auf Seite 306) geschrieben.

Um die Anzahl der Zugriffe auf Cursor-Dateien möglichst gering zu halten, kann der Systemverwalter über eine DBH-Option die Größe des Cursor-Puffers der jeweiligen DBH-Session anpassen und so die Behandlung von Wiedergewinnungsanweisungen optimieren.

Für die Dimensionierung des Cursor-Puffers bietet die DBH-Statistik des SESAM/SQL-Dienstprogramms SESMON Entscheidungshilfen (siehe Handbuch „Datenbankbetrieb“).

8.6.4 Planpuffer

Der SESAM/SQL-DBH reserviert einen Bereich des Hauptspeichers eigens für die Speicherung von SQL-Zugriffsplänen.

SQL-Zugriffsplan

Ein SQL-Zugriffsplan ist eine Auswertungsvorschrift für eine SQL-Anweisung.

Für eine statische SQL-Anweisung wird bei der ersten Ausführung ein SQL-Zugriffsplan erzeugt. Nach erfolgreicher Ausführung wird dieser Plan nicht sofort gelöscht, sondern im Planpuffer abgelegt. Wird die gleiche SQL-Anweisung erneut ausgeführt, steht der zugehörige Plan bereits zur Verfügung und muss nicht nochmals erzeugt werden, was einen erheblichen Performance-Gewinn bedeutet.

Eine dynamische SQL-Anweisung erzeugt einen SQL-Zugriffsplan jeweils bei den Anweisungen EXECUTE IMMEDIATE oder PREPARE (siehe Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“). Bei der Anweisung EXECUTE IMMEDIATE wird der zugehörige Plan unmittelbar erzeugt und sofort ausgeführt. Bei der Anweisung PREPARE bleibt der Plan mindestens bis Transaktionsende im Planpuffer, so dass Folgeanweisungen auf ihn zugreifen können (siehe auch Handbuch „Performance“).

Wird in einer Folgetransaktion dieselbe dynamische SQL-Anweisung mit EXEC, IMMEDIATE oder PREPARE bearbeitet, wird der zugehörige Plan wiederverwendet, sofern er noch im Planpuffer vorhanden ist.

Größe des Planpuffers

Der Planpuffer ist aufgeteilt in einen Primärpuffer, in dem der DBH die SQL-Zugriffspläne speichert und einen Sekundärpuffer, der die zugehörige Verwaltungsinformation enthält. Die Größe des Planpuffers ist abhängig von zwei Faktoren:

- der DBH-Option SQL-SUPPORT mit ihrem Operanden PLANS, der die Mindestanzahl paralleler SQL-Zugriffspläne festlegt
- der DBH-Option COLUMNS, welche die Bereichsgröße für Wiedergewinnungsanweisungen festlegt

Beide Werte kann der Systemverwalter den Bedürfnissen der aktuellen Session anpassen (siehe Handbuch „Datenbankbetrieb“).

Reicht die Größe des Planpuffers nicht aus, so werden die SQL-Zugriffspläne nach dem LRU-Prinzip verdrängt: derjenige Plan, der am längsten nicht mehr verwendet wurde, wird von einem neuen Plan überschrieben.

Informationen über den Planpuffer liefert die Betriebsstatistik des SESAM/SQL-Dienstprogramms SESMON. In der Maske „SQL-INFORMATION“ sind u.a. die Größe des Planpuffers und seine aktuelle Belegung ablesbar.

8.6.5 Transfer-Container

Bei der Bearbeitung von SQL-Zugriffsplänen entstehen sogenannte SQL-Scans. Das sind Teilbereiche einer Auswertungsvorschrift für eine SQL-Anweisung.
Bei der Bearbeitung von CALL-DML-Anwendungen werden für jede logische Datei mit der OPEN-Anweisung Frage- und Antwortbereiche angefordert.

Um für Frage- und Antwortbereiche von SQL-Scans bzw. logischen Dateien Hauptspeicher verfügbar zu haben, führt der SESAM/SQL-DBH den Transfer-Container, der in festen Einheiten von 256 Byte verwaltet wird.

Größe des Transfer-Containers

Zu Beginn der DBH-Session legt der DBH den Transfer-Container in einer bestimmten Größe an. Die Grundeinstellung der Transfer-Container-Größe beträgt 64 Kbyte, kann aber per DBH-Option erweitert werden.
Häufige Speicheranforderungen im Laufe der Session können zu einer starken Fragmentierung des Transfer-Containers führen, so dass der DBH schließlich die aktuellen Anforderungen nicht mehr ausführen kann. In diesem Fall wird der Transfer-Container zunächst reorganisiert. Führt dies nicht zu dem erwünschten Erfolg, so erweitert der DBH den Transfer-Container bis zu einer bestimmten Maximalgröße.

Über eine DBH-Option kann der Systemverwalter die Grundeinstellung der Containergröße sowie die Maximalgröße des Containers bedarfsgerecht dimensionieren. Im laufenden DBH-Betrieb kann er die Maximalgröße mit der Administrationsanweisung MODIFY-STORAGE-SIZE den Anforderungen der aktuellen Session anpassen. Dabei bietet ihm die DBH-Statistik des SESAM/SQL-Dienstprogramms SESMON in der Maske „SYSTEM INFORMATION“ hilfreiche Informationen (siehe Handbuch „Datenbankbetrieb“).

8.6.6 Work-Container

Der SESAM/SQL-DBH analysiert die Anweisungen aus den Anwenderprogrammen auf lexikalische, syntaktische und semantische Korrektheit. Aus einer korrekten Anweisung bzw. aus dem Teilbereich einer SQL-Anweisung erzeugt der DBH ein optimiertes Format, die sogenannte Arbeitsleiste.

Aus einer CALL-DML-Anweisung erzeugt der DBH die Arbeitsleiste direkt.

Aus einer SQL-Anweisung wird zunächst ein SQL-Zugriffsplan erstellt, also eine Auswertungsvorschrift für die SQL-Anweisung. Ein SQL-Zugriffsplan besteht aus mindestens einem, meist aber mehreren Teilbereichen, den sogenannten SQL-Scans. Das optimierte Format eines Scans bildet schließlich die Arbeitsleiste.

Arbeitsleisten werden für die weitere Verarbeitung im Work-Container gespeichert, so dass Folgeanweisungen auf sie zurückgreifen können. Eine Folgeanweisung kann z.B. die SQL-Anweisung FETCH sein aber auch das Fortsetzen der Verarbeitung nach Entsperren einer ehemals gesperrten Transaktion.
Durch die Speicherung von Arbeitsleisten reduziert sich der Analyse- und Optimierungsaufwand erheblich.

Über eine DBH-Option kann der Systemverwalter den Work-Container beim Starten der DBH-Session bedarfsgerecht dimensionieren. Im laufenden DBH-Betrieb kann er die Maximalgröße mit der Administrationsanweisung MODIFY-STORAGE-SIZE den Anforderungen der aktuellen Session anpassen. Die DBH-Statistik des SESAM/SQL-Dienstprogramms SESMON bietet dafür in der Maske „SYSTEM INFORMATION“ hilfreiche Informationen (siehe Handbuch „Datenbankbetrieb“).

8.7 Weitere Mechanismen zur Durchsatzsteigerung

Neben einer differenzierten Puffertechnik (siehe Abschnitt „Puffer und Container des Data Base Handlers“ auf Seite 311) realisiert der SESAM/SQL-DBH noch zahlreiche Mechanismen zur Durchsatzsteigerung. Dazu gehören sowohl interne Optimierungsmaßnahmen als auch solche, in die der Systemverwalter steuernd eingreifen kann.

Die nachfolgenden Abschnitte beschreiben Mechanismen zur Durchsatzsteigerung, die der Systemverwalter beeinflussen kann:

- Multitasking
- Multi-Thread-Betrieb
- Prioritätensteuerung
- Flexible Bearbeitung von Wiedergewinnungsanweisungen
- Auslagerung CPU-intensiver Aktionen

8.7.1 Multitasking

Das Datenbanksystem SESAM/SQL bietet in der Enterprise Edition die Möglichkeit, zur Durchsatzsteigerung eine Multiprozessoranlage dadurch auszunutzen, dass ein DBH mit mehreren Tasks (DBH-Option DBH-TASKS) geladen wird. Damit kann die dem DBH zur Verfügung stehende CPU-Leistung besser genutzt werden.
Die Tasks eines DBH bearbeiten alle einen gemeinsamen Datenbestand.

Der Systemverwalter arbeitet mit einem Single-System-Image, d.h. er administriert den DBH insgesamt und keine bestimmte Task. Alle Ausgaben des laufenden Systems erfolgen bei der Task, die der Systemverwalter selbst gestartet hat und mit dieser Task kommuniziert er auch.

Der DBH verteilt die Last automatisch auf die DBH-Tasks. Die Tasks werden lastabhängig genutzt, d.h. es wird versucht, die Last mit möglichst wenig Tasks zu bewältigen.

Über das SESAM/SQL-Dienstprogramm SESMON kann der Systemverwalter in der Maske „TASKS“ die Auslastung der Tasks beobachten.

8.7.2 Multi-Thread-Betrieb

Das Datenbanksystem SESAM/SQL läuft innerhalb von BS2000 als nicht-privilegiertes Programm ab. Wie bei jeder anderen Anwendertask auch, wird die Verarbeitung des SESAM/SQL-DBH bei verschiedenen Ereignissen unterbrochen, etwa bei Ablauf der Zeitscheibe oder bei Ein-/Ausgabe-Vorgängen mit WAIT.

Asynchrone Ein-/Ausgabe

Die Datenbank-Bearbeitung ist in der Regel sehr ein-/ausgabeintensiv. Damit die Arbeit des DBH nicht zu häufig unterbrochen wird, behandelt SESAM/SQL alle Ein-/Ausgabe-Vorgänge in einer Weise, dass der DBH in den entstehenden Wartezeiten andere Tätigkeiten durchführen kann.

Asynchrone Ein-/Ausgabe, also die Zerlegung von Ein-/Ausgabe-Vorgängen, ist besonders für den Multi-Thread-Betrieb sinnvoll: im Multi-Thread-Betrieb kann der DBH mehrere Aufträge parallel verarbeiten, so dass jeweils ein Auftrag die Zeit nutzen kann, in der ein anderer Auftrag auf den Abschluss eines Ein-/Ausgabe-Vorgangs wartet.

Informationen über den gegenwärtigen Verarbeitungszustand eines Auftrags sind jeweils in speziellen Sicherstellungsbereichen, den Thread-Bereichen, hinterlegt. Mit Hilfe dieser Informationen kann der DBH einen unterbrochenen und wartenden Auftrag nach Durchführung des entsprechenden Ein-/Ausgabe-Vorgangs weiterverarbeiten.

Behandeln von Wartesituationen

Wird ein Auftrag wegen Anstoß eines Ein-/Ausgabe-Vorgangs unterbrochen, so sind die zugehörigen Datenbestände im Allgemeinen logisch und physikalisch inkonsistent. Durch geeignete Sperren verhindert der DBH einen Zugriff paralleler Aufträge auf diese Datenbestände.
Die Bearbeitung eines Auftrags wird unterbrochen, wenn es zu einem Konflikt bei einer solchen Sperre oder einer Transaktionssperre kommt.
Ebenso wird die Bearbeitung eines Auftrags im DBH unterbrochen, wenn gerade eine Service-Task für diesen Auftrag arbeitet.
Die Unterbrechung eines Auftrags im DBH bedeutet, dass der zugehörige Thread wartet, bis seine benötigten Betriebsmittel verfügbar sind.

Multi-Thread-Betrieb steuern

Über die DBH-Option THREADS kann der Systemverwalter beim Starten des independent DBH den Multi-Thread-Betrieb aktivieren.

Der independent DBH kann bis zu 1024 Threads verwalten. Dabei nutzt er die Threads lastabhängig, d.h. nicht alle Threads sind ständig belegt, was sich positiv auf den Verwaltungsaufwand auswirkt.

Beim linked-in DBH wird stets nur ein Thread generiert.

Über die DBH-Option THREADS legt der Systemverwalter auch die maximale Anzahl parallel aktiver Threads fest.

Der Wert der DBH-Option THREADS wird intern mit der Option DBH-TASKS verglichen. Er muss beim independent DBH größer/gleich 2 * der Anzahl der Tasks sein.
Beim linked-in DBH ist die DBH-Option THREADS auf den Wert 1 gesetzt.

8.7.3 Prioritätensteuerung

Der SESAM/SQL-DBH bearbeitet normalerweise Aufträge mit verschiedenen BS2000-Prioritäten in der Reihenfolge ihres Eintreffens. Der Systemverwalter kann diese Arbeitsweise modifizieren, indem er über die DBH-Option REQUEST-CONTROL beim Starten des independent DBH die Prioritätensteuerung aktiviert. Beim linked-in DBH steht die Prioritätensteuerung nicht zur Verfügung.

Prioritätsklassen und Gewichte

Über die Prioritätensteuerung des independent DBH ist es möglich, BS2000-Prioritäten für Aufträge zu relativieren und den Bedürfnissen der DBH-Session anzupassen. Dafür unterteilt der Systemverwalter den gesamten Bereich an BS2000-Prioritäten (Priorität 30 bis 255) in drei Prioritätsklassen. Durch Vergabe eines Gewichts für jede Prioritätsklasse kann er die Abarbeitungsreihenfolge für Aufträge der einzelnen Prioritätsklassen bedarfsgerecht steuern.

Eingreifen im laufenden Betrieb

Bei Bedarf kann der Systemverwalter auch während der laufenden Session in die Prioritätensteuerung eingreifen, wofür ihm verschiedene Administrationsanweisungen zur Verfügung stehen (siehe Abschnitt „Administrationsanweisungen und -kommandos“ auf Seite 328).

8.7.4 Flexible Bearbeitung von Wiedergewinnungsanweisungen

Die Bearbeitung von Wiedergewinnungsanweisungen kann u. U. sehr lange dauern und andere Aufträge blockieren. Damit die Performance nicht unnötig belastet wird, passt der DBH in solchen Fällen seine Suchstrategie den jeweiligen Bedingungen an.

Bearbeitung über Sekundärindizes

Der DBH unterbricht die Bearbeitung von Wiedergewinnungsanweisungen über Sekundärindizes und setzt sie sequenziell fort, wenn die Suche über Sekundärindizes aus Sicht des DBH keinen Vorteil bietet.

Sequenzielle Bearbeitung

Der DBH unterbricht die sequenzielle Bearbeitung von Wiedergewinnungsanweisungen, sobald die Anzahl der logischen Zugriffe auf Anwenderdaten einen festgesetzten Grenzwert überschreitet (siehe Handbuch „Datenbankbetrieb“).
Anweisungen, die aus diesem Grund unterbrochen wurden, werden bis zur Fortsetzung der Bearbeitung in eine Warteschlange eingereiht. Der Auftrag bleibt fest mit dem Thread verbunden. Der Thread steht nicht für andere Aufträge zur Verfügung.

Ist die Anzahl logischer Zugriffe auf Anwenderdaten aus Sicht des DBH unvertretbar hoch, überschreitet sie also einen hierfür festgesetzten Grenzwert, so bricht der DBH die Bearbeitung vollständig ab.

Bearbeitung von Wiedergewinnungsanweisungen steuern

Über die DBH-Option RETRIEVAL-CONTROL kann der Systemverwalter die Kriterien, nach denen der DBH seine Suchstrategie wählt, den Bedürfnissen der Session anpassen. So kann er z.B. den Grenzwert für die Anzahl logischer Zugriffe auf Anwenderdaten festlegen, bei dessen Erreichen der DBH die sequenzielle Bearbeitung abbrechen soll. Auch während der laufenden DBH-Session stehen ihm Administrationsanweisungen zur Verfügung, um die Suchstrategie des DBH bedarfsgerecht zu steuern (siehe Abschnitt „Administrationsanweisungen und -kommandos“ auf Seite 328).

8.7.5 Auslagerung CPU-intensiver Aktionen

CPU-intensive Aktionen, etwa das Sortieren von Zwischenergebnismengen, können sehr lange dauern. Damit der SESAM/SQL-DBH während dieser Zeit nicht blockiert ist, lagert er solche CPU-intensiven Aktionen aus in eigene Tasks, die für Service-Aufträge zur Verfügung stehen.

Service-Tasks stehen beispielsweise für die Sortierung von Zwischenergebnismengen zur Verfügung, aber auch für einige Utility-Funktionen (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“). So werden die Utility-Anweisungen COPY, LOAD OFFLINE und UNLOAD OFFLINE automatisch in Service-Tasks ausgelagert, aber auch Teile von Utility-Funktionen, wie das Einspielen oder Anpassen eines SESAM-Sicherungsbestands, der Indexaufbau oder die Reorganisation von Anwenderdaten.

Eine Service-Task ist nicht jeweils einer Auftragsart zugeordnet, sondern kann alle Service-Aufträge ausführen. Dadurch ist es möglich, die Anzahl zur Verfügung stehender Service-Tasks gering zu halten und eine optimale Ausnutzung der Tasks zu erzielen.

Bei Session-Ende werden alle Service-Tasks vom DBH beendet. Noch laufende oder offene Service-Aufträge werden abgebrochen.

Auslastung von Service-Tasks steuern

Wieviele Tasks zu Beginn einer DBH-Session verfügbar sein sollen und wieviele Tasks im Laufe der Session maximal gestartet werden dürfen, kann der Systemverwalter über die DBH-Option SERVICE-TASKS sessionspezifisch festlegen. Beim Linked-in DBH steht allerdings höchstens eine Service-Task zur Verfügung.

Über das SESAM/SQL-Dienstprogramm SESMON kann der Systemverwalter in der Maske „SERVICE TASKS“ die Auslastung der Service-Tasks beobachten.

Mit der Administrationsanweisung MODIFY-SERVICE-TASKS kann der Systemverwalter Anzahl und Attribute der Service-Tasks im laufenden Betrieb anpassen.

8.8 Startkommandos für SESAM/SQL-Programme

Alle SESAM/SQL-Programme werden über SESAM-Startkommandos gestartet.

Dabei wird davon ausgegangen, dass Sie SESAM/SQL und CRTE mit IMON installiert oder die CRTE- und SESAM-Bibliotheken unter den auf Seite 322 beschriebenen Standard-Dateinamen abgelegt haben.

Die Startkommandos für alle SESAM/SQL-Programme sind in der ausgelieferten SDF-Syntaxdatei von SESAM/SQL-Server definiert (siehe auch Handbuch „Datenbankbetrieb“):

Startkommando	**Programm**
START-SESAM-DBH	SESAM
START-SESAM-DCN	Verteilkomponente SESDCN
START-SESAM-ADMINISTRATION	Administrationsprogramm SESADM
START-SESAM-PERF-MONITOR	SESMON
START-SESAM-TUNING-TRACE-EVAL	SESCOSP
START-SESAM-RETRIEVAL-DIALOGUE	SEDI61
START-SESAM-CDML-DIALOGUE	SEDI63
START-SESAM-LOG-FILE-EVAL	SEDI70
START-SESAM-UTILITY-MONITOR	Utility-Monitor SESUTI

Tabelle 58: Startkommandos

Die Startkommandos werden in den jeweiligen Beschreibungen zum Starten der Programme beschrieben.

Das Kommando START-PROGRAM kann aus Kompatibilitätsgründen weiterhin angewendet werden. Informationen dazu finden Sie im Anhang des Handbuchs „Datenbankbetrieb“.

Die Startkommandos übernehmen folgende Aufgaben:

- Ermitteln und Zuweisen der Systemdateien
  Die Startkommandos ermitteln die Systemdateien, die für den Start des Programms benötigt werden und weisen die Dateien über entsprechende Linknamen zu.
  Die Namen der benötigten Systemdateien werden nach folgendem Algorithmus ermittelt: Es wird versucht über IMON den Namen der Systemdatei zu ermitteln.
  Von SESAM/SQL werden die Dateien der aktuellen Version gesucht. Von CRTE werden die Dateien der aktuellsten installierten Version gesucht. Gelingt dies nicht, so wird der Standardname dieser Datei verwendet.
  Die folgende Liste zeigt die betroffenen Dateien und die dazugehörigen Standardnamen:

Datei	Linkname	Standardname
CRTE-Bibliothek	BLSLIBxx	$.SYSLNK.CRTE für /390-Server $.SKULNK.CRTE für x86-Server
SESAM-Modulbibliothek	SESAMOML	$.SYSLNK.SESAM-SQL.<ver> für /390-Server $.SKULNK.SESAM-SQL.<ver> für x86-Server
Prozedur für Kommando CONNECT-SESAM-CONFIGURATION		$.SYSSPR.SESAM-SQL.<ver>.RUN-CFG
Prozedur für die Startkommandos		$.SYSSPR.SESAM-SQL.<ver>.RUN-STA
FHS-Formatbibliothek für SESMON	MAPLIB	$.SYSFHS.SESAM-SQL.<ver>.MON.E
FHS-Formatbibliothek für den Utility-Monitor (deutsch/englisch)	MAPLIB	$.SYSFHS.SESAM-SQL.<ver>.UTI.D
	MAPLIB	$.SYSFHS.SESAM-SQL.<ver>.UTI.E
Hilfetextdatei für den Utility-Monitor (deutsch/englisch)	SEEHELP	$.SYSMAN.SESAM-SQL.<ver>.UTI.D
	SEEHELP	$.SYSMAN.SESAM-SQL.<ver>.UTI.E

Tabelle 59: Standardnamen der Systemdateien

- Bestimmen der Betriebsart
  Mit den Startkommandos werden die Programme grundsätzlich im Dialog gestartet. Wünscht der Anwender, dass ein Programm im Batchbetrieb ablaufen soll, so muss er das Startkommando in einem ENTER-Job aufrufen.

8.9 Steuerung und Überwachung der Session

Bereits beim Starten des DBH, ebenso wie beim Starten von SESDCN, kann der Systemverwalter die jeweilige Komponente bedarfsgerecht parametrisieren. Er kann aber auch die laufende Session überwachen und ggf. steuernd in den Betrieb eingreifen.

Die folgenden Abschnitte beschreiben wichtige Funktionen zur Steuerung und Überwachung der DBH- und der SESDCN-Session:

- DBH-Startanweisungen und -Optionen
- SESDCN-Steueranweisungen und DCN-Optionen
- Adminstrationsanweisungen und -kommandos für den DBH und für SESDCN
- E-Mail-Versand wichtiger Informationen der DBH-Session
- Auftragsprotokollierung mit SESCOSP
- Auswertung des SESAM/SQL-Betriebs mit SESMON
- DA-LOG-Aufbereitung durch SEDI70

8.9.1 DBH-Startanweisungen und -Optionen

DBH-Startanweisungen und -Optionen dienen der Parametrisierung des DBH. Sie legen für die aktuelle DBH-Session Betriebsmittel, Grenzwerte und Arbeitsregeln fest.

Eingabe der DBH-Startparameter

Die DBH-Startanweisungen gibt der Systemverwalter beim Starten des SESAM/SQL-DBH in einer bestimmten Reihenfolge ein. Nach der ersten Startanweisung, SET-DBH-OPTIONS, fügt er je nach Bedarf die DBH-Optionen ein. Der DBH liest die Startanweisungen und -Optionen von SYSDTA.

Für die Eingabe der DBH-Startanweisungen und -Optionen gibt es folgende Möglichkeiten:

- Eingabe innerhalb der Prozedur, mit welcher der DBH gestartet wird
- Eingabe über eine Datei, der der Systemverwalter vor dem Starten des DBH die Systemdatei SYSDTA zuweist
- Eingabe direkt am Bildschirm (im Dialogbetrieb)
- Eingabe über die Konfigurationsdatei, die der Systemverwalter über den Linknamen SESCONF oder die Anweisung CONNECT-SESAM-CONFIGURATION zuweist

Die Formate der DBH-Startanweisungen und -Optionen entsprechen den Regeln von SDF (**S**ystem **D**ialog **F**acility, siehe Handbuch „Dialogschnittstelle SDF“). SDF unterstützt die Eingabe der DBH-Startanweisungen und Optionen im geführten Dialog, analysiert eine eingegebene Anweisung und übergibt sie zur Weiterverarbeitung an den DBH.

DBH-Startanweisungen

DBH-Startanweisungen leiten die Parametrisierung des DBH ein. Sie veranlassen das Einlesen der DBH-Optionen sowie das Einfügen von Einträgen in das SQL-Datenbankverzeichnis bzw. in das CALL-DML-Tabellenverzeichnis. Die Eingabe der DBH-Startanweisungen gliedert sich in drei SDF-Anweisungen:

1. SET-DBH-OPTIONS
2. ADD-SQL-DATABASE-CATALOG-LIST
3. ADD-OLD-TABLE-CATALOG-LIST

Die erste DBH-Startanweisung muss die SET-DBH-OPTIONS-Anweisung sein. Die übrigen beiden Anweisungen sind optional, da Einträge in das SQL-Datenbankverzeichnis bzw. CALL-DML-Tabellenverzeichnis auch über Administrationsanweisungen erfolgen können.

DBH-Optionen

DBH-Optionen parametrisieren den DBH und definieren so die wesentlichen Merkmale einer Session. Die meisten DBH-Optionen (Ausnahmen: CONFIGURATON-NAME und DBH-NAME) kann der Systemverwalter im laufenden Betrieb ändern. Die aktuellen DBH-Optionen kann der Systemverwalter in eine Datei sichern und in der nächsten DBH-Session wieder verwenden.

DBH-Optionen lassen sich untergliedern in übergeordnete DBH-Optionen und untergeordnete DBH-Optionen.

- Übergeordnete DBH-Optionen, wie ADMINISTRATION oder STORAGE SIZE, beziehen sich jeweils auf einen bestimmten Themenkomplex.
- Untergeordnete DBH-Optionen behandeln jeweils einen Teilbereich des Themenkomplexes der zugehörigen übergeordneten Option.

Allen übergeordneten DBH-Optionen sind eine oder mehrere untergeordnete Optionen zugeordnet. Der Systemverwalter gibt die DBH-Optionen nach Eingabe der DBH-Startanweisung SET-DBH-OPTIONS ein.

Für alle DBH-Optionen gibt es Standardeinstellungen. Der Systemverwalter muss daher nur solche Optionen explizit angeben, für die er von der Standardeinstellung abweichende Werte wünscht. Will der Systemverwalter für eine oder mehrere untergeordnete Optionen

eigene Werte vergeben, so muss er zunächst die zugehörige übergeordnete Option eintragen. Viele DBH-Optionen kann der Systemverwalter im laufenden Betrieb mit Administrationsanweisungen (MODIFY-...) ändern.

Welche DBH-Optionen dem Systemverwalter zur Verfügung stehen, zeigt die Übersicht auf den folgenden Seiten. Eine detaillierte Beschreibung der DBH-Optionen finden Sie im Handbuch „Datenbankbetrieb“.

Bei der Eingabe von DBH-Optionen über SYSDTA müssen Kommentarzeilen mit „//REMARK“ eingeleitet werden. Die Angabe „/REMARK“ oder „REMARK“ wird als Kommando interpretiert und führt zu einer SDF-Fehlermeldung. Kommentare dürfen nicht innerhalb von Anweisungen stehen.

Übersicht der DBH-Optionen

übergeordnete Option	untergeordnete Option	**Kurzbeschreibung**
DBH-IDENTIFICATION	CONFIGURATION-NAME DBH-NAME	**DBH identifizieren** Konfigurationsnamen festlegen DBH-Namen vergeben
ADMINISTRATION	ACCOUNTING ADMINISTRATOR MSG-OUTPUT SECURITY	**DBH administrieren** Abrechnung aktivieren Administrationsberechtigung vergeben Ausgaben des DBH steuern unberechtigten Zugriff verhindern
CPU-RESOURCES	DBH-TASKS SERVICE-TASKS	**CPU-Auslastung steuern** Anzahl der DBH-Tasks in der Session Anzahl der Tasks für Service-Aufträge festlegen
FILE-RESOURCES	MEDIA-CATALOG SESSION-LOGGING-ID	**Vereinbarungen zu Dateien treffen** Medienkatalog anlegen sessionbezogene Dateien kennzeichnen
LINKED-IN-ATTRIBUTES	CODED-CHARACTER-SET	**Attribute des linked-in DBH angeben** Codierten Zeichensatz angeben, mit dem das Anwenderprogramm arbeitet.
RECOVER-OPTIONS	SYSTEM-DATA-BUFFER USER-DATA-BUFFER MEDIA-CATALOG	**Vereinbarungen für einen RECOVER- oder REFRESH-Lauf treffen** System-Data-Buffer dimensionieren User-Data-Buffer dimensionieren Medienkatalog anlegen

Tabelle 60: Über- und untergeordnete DBH-Optionen (Teil 1 von 2)

übergeordnete Option	untergeordnete Option	**Kurzbeschreibung**
STORAGE-SIZE		**Puffer- und Containergröße einstellen**
	CURSOR-BUFFER	Cursor-Puffer dimensionieren
	SYSTEM-DATA-BUFFER	System-Data-Buffer dimensionieren
	TRANSFER-CONTAINER	Transfer-Container dimensionieren
	USER-DATA-BUFFER	User-Data-Buffer dimensionieren
	WORK-CONTAINER	Work-Container dimensionieren
SYSTEM-LIMITS		**Grenzwerte einstellen**
	COLUMNS	Bereich für Wiedergewinnungsanweisungen vergrößern
	OLD-TABLE-CATALOG	maximale Anzahl Einträge im CALL-DML-Tabellenverzeichnis festlegen
	SPACES	maximale Anzahl gleichzeitig zugreifbarer Spaces festlegen
	SQL-DATABASE-CATALOG	maximale Anzahl Einträge im SQL-Datenbankverzeichnis festlegen
	SQL-SUPPORT	Grenzwerte der SQL-Schnittstelle festlegen
	SUBORDERS	SQL-Scans bzw. logische Dateien zur Verfügung stellen
	SYSTEM-THREADS	maximale Anzahl paralleler Systemthreads festlegen
	THREADS	maximale Anzahl paralleler Threads festlegen
	USERS	maximale Anzahl paralleler Auftraggeber festlegen
SYSTEM-STRATEGIES		**Verarbeitungsstrategie festlegen**
	REQUEST-CONTROL	Verdrängungsmodus für Puffer festlegen
	RESTART-CONTROL	Zeitdauer der Verfügbarkeit bei Wiederanlauf steuern
	RETRIEVAL-CONTROL	Prioritätensteuerung einsetzen
	TRANSACTION-SECURITY	Suchstrategie des DBH beeinflussen Transaktionssicherung aktivieren

Tabelle 60: Über- und untergeordnete DBH-Optionen (Teil 2 von 2)

8.9.2 SESDCN-Steueranweisungen und Optionen

SESDCN-Steueranweisungen und -Optionen parametrisieren die Verteilkomponente SESDCN und definieren die Verteilregel (siehe Abschnitt „Verteilte Verarbeitung mit SESAM/SQL-DCN“ auf Seite 283).

SESDCN-Steueranweisungen

Der Systemverwalter gibt die SESDCN-Steueranweisungen beim Starten von SESDCN ein. Die Eingabe erfolgt über die gleichen Schnittstellen wie die Eingabe der DBH-Startanweisungen und -Optionen (siehe „Eingabe der DBH-Startparameter“ auf Seite 323).

Die Eingabe der SESDCN-Steueranweisungen gliedert sich in drei SDF-Anweisungen:

1. SET-DCN-OPTIONS
2. ADD-DISTRIBUTION-RULE-LIST
3. ADD-NETWORK-LINK-LIST

Die erste Anweisung veranlasst das Einlesen der DCN-Optionen. Die beiden folgenden Anweisungen leiten die Definition der Verteilregel ein. Die Verteilregel ordnet jeder Datenbank, die an der Verteilung teilnimmt, einen „Zugriffspfad“ zu. Der Zugriffspfad besteht jeweils aus dem Rechnernamen, dem Konfigurationsnamen, dem DCN-Namen und dem DBH-Namen.

DCN-Optionen

Der Systemverwalter gibt die DCN-Optionen unmittelbar nach Eingabe der SESDCN-Steueranweisung SET-DCN-OPTIONS ein.

DCN-Optionen definieren die wesentlichen Merkmale des SESDCN-Betriebs. Da alle DCN-Optionen mit Standardwerten vorbesetzt sind, muss der Systemverwalter nur solche Optionen berücksichtigen, für die er von der Standardeinstellung abweichende Werte braucht.

Die folgende Übersicht bietet eine Kurzbeschreibung der DCN-Optionen. Eine detaillierte Beschreibung der DCN-Optionen finden Sie im Handbuch „Datenbankbetrieb“.

Bei der Eingabe von DCN-Optionen über SYSDTA müssen Kommentarzeilen mit „//REMARK“ eingeleitet werden. Die Angabe „/REMARK“ oder „REMARK“ wird als Kommando interpretiert und führt zu einer SDF-Fehlermeldung. Kommentare dürfen nicht innerhalb von Anweisungen stehen.

Übersicht über die DCN-Optionen

DCN-OPTION	Kurzbeschreibung der Funktion
ADMINISTRATOR	Administrationsberechtigung vergeben
COLDSTART	Kaltstart anfordern
DCN-IDENTIFICATION	DCN-Namen und Konfigurationsnamen vergeben
REMOTE-ACCESS	Remote-Zugriff von entferntem Rechner aus erlauben
SESDLG-PASSWORD	Kennwort zuweisen
SYSTEM-LIMITS	Grenzwerte für die zugelassene Anzahl Auftraggeber und Teilhaber-Anwendungen festlegen sowie Rücksetzkriterien für Transaktionen vereinbaren

Tabelle 61: DCN-Optionen

8.9.3 Administrationsanweisungen und -kommandos

Um in den laufenden Betrieb eingreifen zu können, stehen dem Systemverwalter Administrationsanweisungen und -kommandos zur Verfügung:

- mit Hilfe DBH-spezifischer Administrationsanweisungen und -kommandos kann der Systemverwalter die DBH-Session administrieren
- bei der verteilten Verarbeitung mit SESAM/SQL-DCN (siehe Seite 283) kann der Systemverwalter mit SESDCN-spezifischen Administrationanweisungen und -kommandos die Verteilkomponente SESDCN administrieren.

Will der Systemverwalter gezielt Informationen zum Datenbankbetrieb abfragen, so steht ihm das SESAM/SQL-Dienstprogramm SESMON zur Verfügung. Die DBH-Statistik von SESMON bietet Informationen, mit deren Hilfe der Systemverwalter die DBH-Session überwachen und die optimale Einstellung der DBH-Optionen ermitteln kann. Informationen zum SESDCN-Betrieb bietet die SESDCN-Statistik (siehe Abschnitt „Betriebsdaten ausgeben mit SESMON“ auf Seite 336).

Eingabe der Administrationsanweisungen und -kommandos

Für die Eingabe der Administrationsanweisungen bzw. -kommandos gibt es folgende Möglichkeiten:

- über das Administrationsprogramm SESADM, das als eigener Prozess läuft, aber auch als Unterprogramm des Utility-Monitors verfügbar ist
- über das BS2000-Kommando INFORM-PROGRAM
- über CALL-DML-Anweisungen innerhalb eines CALL-DML-Programms

Zugang zum Administrationsprogramm SESADM erhalten Sie auch aus dem World Wide Web, siehe Seite 38.
Die Bedienung von SESADM ist im Handbuch „Datenbankbetrieb“ beschrieben.

Eingabe-Format

Je nachdem, über welche Schnittstelle die Administration erfolgen soll, muss der Systemverwalter Administrationsanweisungen oder Administrationskommandos eingeben.

- Administrationsanweisungen stehen für die Administration über SESADM zur Verfügung. Das Administrationsprogramm SESADM liest die Eingaben mit Hilfe der Dialogschnittstelle SDF ein. Die Syntax der Administrationsanweisungen folgt daher den Regeln von SDF (z.B. SHOW-INACTIVE-SQL-USERS).
- Bei der Administration über INFORM-PROGRAM bzw. innerhalb eines CALL-DML-Programms stehen dem Systemverwalter Administrationskommandos im ISP-Format zur Verfügung (z.B. USER,INACT).

Administrationsanweisungen bieten nahezu den gleichen Funktionsumfang wie Administrationskommandos. In den folgenden Übersichten werden daher nur Administrationsanweisungen berücksichtigt. Eine Kurzbeschreibung der entsprechenden Administrationskommandos finden Sie im Handbuch „Datenbankbetrieb“.

Ausgabe der Administrationsanweisungen und -kommandos

Große Datenbankkonfigurationenen liefern z.B. in den SHOW-Anweisungen von SESADM auch umfangreiche Ausgaben. Mit der Anweisung MODIFY-OUTPUT-MODE im DBH- und im DCN-Menü können Sie die Ausgabe von Administrationsanweisungen auf SYSOUT/SYSLST oder in eine temporäre Datei steuern.
Die Informationsausgaben der meisten SHOW-Anweisungen können in S-Variablen ausgegeben und in S-Prozeduren weiter verarbeitet werden.

Unter SESADM sind alle Ausgaben vollständig. Bei Administration über INFORM-PROGRAM oder ein CALL-DML-Programm hat ein Ausgabeabschnitt (eine sogenannte Anwort) eine maximale Größe von 32000 Byte. Größere Ausgabemengen werden in mehrere Antworten aufgeteilt. Die Folgeantworten können Sie mit dem Administrationskommando NEXT abrufen.

Resultat der Administrationsanweisungen

Rückmeldungen und Fehlermeldungen werden bei einem Ablauf im Dialog nach SYSOUT und SYSLST, bei einem Ablauf im Batchbetrieb auf die Bedienstation und nach SYSLST ausgegeben. Zusätzlich wird die Meldungsnummer der letzten Antwort des DBH oder von SESDCN auf eine Administrationsanweisung in der S-Variablen SESAM-RESULT und in der Jobvariablen #SESAM.SESADM.JV vermerkt. Dies erlaubt eine automatisierte Administration mit SESADM. Siehe Handbuch „Datenbankbetrieb“.

Administrationsanweisungen des DBH

Die folgenden Tabellen zeigen eine Übersicht der Administrationsanweisungen des DBH.

Die ausführliche Beschreibung von Syntax und Bedeutung der DBH-Administrationsanweisungen finden Sie im Handbuch „Datenbankbetrieb“. Dort finden Sie auch eine zuordnende Gegenüberstellung von Administrationsanweisungen und -kommandos.

Die drei Tabellen sind nach folgenden Gesichtspunkten aufgeteilt:

- Tabelle 62 beschreibt alle Administrationsanweisungen, die für den Auftraggeber spezifische Auskünfte geben
- Tabelle 63 fasst alle Administrationsanweisungen zusammen, die die eingestellten DBH-Startanweisungen und -Optionen anzeigen oder ändern (siehe Abschnitt „Steuerung und Überwachung der Session“ auf Seite 323)
- Tabelle 64 zeigt alle übrigen Anweisungen, die steuernd in den SESAM/SQL-Betrieb eingreifen.

Administrationsanweisung	Kurzbeschreibung
SHOW-USER-SPACES	Ausgabe der vom Anwender genutzten und damit gesperrten Spaces
SHOW-CALL-DML-SUBORDERS	Anzahl der CALL-DML-OPEN-Aufträge ausgewählter Auftraggeber ausgeben
SHOW-CATALOG-USERS	Anzahl aktiver Auftraggeber zu ausgewählten Datenbanken ausgeben
SHOW-INACTIVE-SQL-USERS	alle inaktiven SQL-Auftraggeber ausgeben
SHOW-SPACE-USERS	alle aktiven Auftraggeber eines ausgewählten Space mit entsprechender Zusatzinformation ausgeben
SHOW-TRANSACTIONS	alle offenen Transaktionen ausgewählter Auftraggeber mit entsprechender Zusatzinformation ausgeben
SHOW-USERS	alle aktiven Auftraggeber mit entsprechender Zusatzinformation ausgeben
SHOW-CATID-LIST	Ausgabe der aktuellen CATID-Liste

Tabelle 62: DBH-Administrationsanweisungen für auftraggeber-spezifische Auskünfte

Administrationsanweisung	Kurzbeschreibung
ADD-OLD-TABLE-CATALOG-ENTRY	Eintrag in das CALL-DML-Tabellenverzeichnis hinzufügen
ADD-SQL-DB-CATALOG-ENTRY	Eintrag in das SQL-Datenbankverzeichnis hinzufügen
MODIFY-ADMINISTRATION	Administrationsberechtigung ändern
MODIFY-CATALOG-ACCESS-RIGHTS	Zugriffsrechte für benannte Datenbank ändern
MODIFY-REQUEST-CONTROL	Parameter für die Prioritätensteuerung ändern
MODIFY-RESTART-CONTROL	beeinflusst die Dauer eines möglichen Wiederanlaufs
MODIFY-RETRIEVAL-CONTROL	Kriterium für die Unterbrechung und den Abbruch von Wiedergewinnungsanweisungen ändern
MODIFY-SQL-SORT-LIMIT	Grenzwert modifizieren für die Anzahl Sort-Treffer, die eine Cursortabelle enthalten darf
MODIFY-SUBORDER-LIMIT	Grenzwert modifizieren für die maximale Anzahl SQL-Scans und/oder logischer Dateien von CALL-DML-Aufträgen
MODIFY-MSG-OUTPUT	Ausgabe des DBH ändern
MODIFY-OLD-TABLE-CATALOG-LIMIT	Maximalzahl der Einträge im CALL-DML-Tabellenverzeichnis ändern
MODIFY-RECOVER-OPTIONS	Optionen für nachfolgende RECOVER- oder REFRESH-Läufe ändern
MODIFY-SECURITY	Maximalzahl erlaubter Kennwortverstöße ändern
MODIFY-SERVICE-TASKS	Anzahl und Attribute der Service-Tasks ändern
MODIFY-SESSION-LOGGING-ID	Kennzeichnung session-bezogener Dateien ändern
MODIFY-STORAGE-SIZE	Maximalgröße von Transfer- und Work-Container ändern
MODIFY-TRANSACTION-SECURITY	Parameter der Transaktionssicherung ändern
RECONFIGURE-DBH-SESSION	DBH-Optionen dynamisch ändern
RELOAD-DBH-SESSION	DBH-Module neu laden
REMOVE-OLD-TABLE-CATALOG-ENTRY	Eintrag aus CALL-DML-Tabellenverzeichnis löschen
REMOVE-SQL-DB-CATALOG-ENTRY	Eintrag aus SQL-Datenbankverzeichnis löschen
REUSE-OLD-TABLE-CATALOG-ENTRY	Gültigen Verweis auf einen bereits vorhandenen Tabelleneintrag im CALL-DML-Datenbankverzeichnis herstellen
REUSE-PARTITIONS	Verfügbarkeit von Partitionen wiederherstellen
SAVE-DBH-OPTIONS	Sichern der aktuellen DBH-Optionen

Tabelle 63: Administrationsanweisungen für DBH-Startanweisungen und-Optionen (Teil 1 von 2)

Administrationsanweisung	Kurzbeschreibung
SET-ACCOUNTING-PARAMETER	Protokollierung der Auftragsabrechnung für das RAV-Verfahren steuern
SET-REQUEST-CONTROL	Prioritätensteuerung ein- oder ausschalten
SET-USER-INACTIVE-TIME	Zeitspanne festlegen, nach der offene, aber inaktive Transaktionen eines Auftraggebers zurückgesetzt werden
SHOW-DBH-MEDIA-CATALOG	aktuell gültige DBH-Option MEDIA-CATALOG ausgeben
SHOW-DBH-OPTIONS	aktuell gültige DBH-Optionen ausgeben, mit Ausnahme der DBH-Option MEDIA-CATALOG
SHOW-OLD-TABLE-CATALOG-ENTRIES	Einträge im CALL-DML-Tabellenverzeichnis anzeigen
SHOW-PARTITIONS	Verfügbarkeit von Partitionen ausgeben
SHOW-SQL-DB-CATALOG-ENTRIES	Einträge im SQL-Datenbankverzeichnis anzeigen

Tabelle 63: Administrationsanweisungen für DBH-Startanweisungen und-Optionen (Teil 2 von 2)

Administrationsanweisung	Kurzbeschreibung
ABORT-LOCK-SEQUENCE	Locksequenz eines anderen Auftraggebers beenden
ASSIGN-SYSLST	SYSLST-Datei wechseln
BEGIN-LOCK-SEQUENCE	Locksequenz beginnen
CANCEL-STATEMENT	DML-Anweisung abbrechen
CHANGE-CATLOG	CAT-LOG-Datei und DA-LOG-Dateien wechseln
CHANGE-DALOG	DA-LOG-Dateien wechseln
CLOSE-SPACE	Anwender-Space physikalisch schließen
COMMIT-PTC-TRANSACTION	PREPARE-TO-COMMIT-Transaktion beenden und Transaktion festschreiben
CREATE-DUMP	Hauptspeicherauszug (Dump) erzeugen
END-FOREIGN-COPY	Zustand „copy pending“ auf Spaces nach Fremdkopie austragen.
END-LOCK-SEQUENCE	Locksequenz beenden
HOLD-TRANSACTION-ADMISSION	keine weiteren Transaktionen zulassen
MODIFY-CATID-LIST	aktualisiert die CATID-Liste
PREPARE-FOREIGN-COPY	Schließen einer Datenbank, um eine Fremdkopie erzeugen zu können

Tabelle 64: Weitere DBH-Administrationsanweisungen, die den Datenbankbetrieb steuern (Teil 1 von 2)

Administrationsanweisung	**Kurzbeschreibung**
RELEASE-USER-RESOURCES	alle Betriebsmittel eines Auftraggebers zurücksetzen
RESUME-TRANSACTION-ADMISSION	Transaktionsbetrieb wieder zulassen
ROLLBACK-PTC-TRANSACTION	PREPARE-TO-COMMIT-Transaktion eines Auftraggebers zurücksetzen
ROLLBACK-TRANSACTION	Transaktion eines Auftraggebers zurücksetzen
SET-DBH-MSG-TRACE	Protokollierung von Nachrichten des DBH steuern
SET-DIAGNOSIS-DUMP-PARAMETER	Erstellung eines Dump steuern
SET-SESSION-DIAGNOSIS	Deadlock-Analyse steuern
SET-SQL-DB-CATALOG-STATUS	Status einer Datenbank ändern
SET-TUNING-TRACE	Auftragsprotokollierung steuern
STOP-DBH	DBH-Session beenden

Tabelle 64: Weitere DBH-Administrationsanweisungen, die den Datenbankbetrieb steuern (Teil 2 von 2)

Administrationsanweisungen von SESDCN

Die folgende Tabelle zeigt eine Übersicht aller Administrationsanweisungen, die für die Administration von SESDCN zur Verfügung stehen. Die ausführliche Beschreibung von Syntax und Bedeutung der SESDCN-Administrationsanweisungen finden Sie im Handbuch „Datenbankbetrieb“. Dort finden Sie auch die Zuordnung der Administrationsanweisungen zu den entsprechenden Administrationskommandos.

Administrationsanweisung	Kurzbeschreibung
ADD-DISTRIBUTION-RULE-ENTRY	weitere Datenbank in die Verteilregel eintragen
CREATE-DUMP	Hauptspeicherauszug (Dump) erzeugen
HOLD-TRANSACTION-ADMISSION	keine weiteren Transaktionen zulassen
HOLD-USER-ADMISSION	keine weiteren Auftraggeber zulassen
MODIFY-ADMINISTRATION	Administrationsberechtigung ändern
MODIFY-DISTRIBUTION-RULE-ENTRY	Rechnernamen in der Verteilregel ändern
REMOVE-DISTRIBUTION-RULE-ENTRY	Datenbankeintrag bzw. -einträge aus der Verteilregel löschen
RESUME-TRANSACTION-ADMISSION	Transaktionsbetrieb wieder zulassen
RESUME-USER-ADMISSION	neue Auftraggeber wieder zulassen
ROLLBACK-TRANSACTION	Transaktion eines Auftraggebers zurücksetzen
SET-USER-CALL-TRACE	Protokollierung der Anweisungen ausgewählter Auftraggeber steuern
SET-USER-MSG-TRACE	Protokollierung von SESAM-Nachrichten ausgewählter Auftraggeber steuern
SHOW-DISTRIBUTION-RULE-ENTRIES	aktive Datenbanken der Verteilregel anzeigen
SHOW-TRANSACTIONS	alle offenen Transaktionen ausgewählter Auftraggeber mit entsprechender Zusatzinformation ausgeben
SHOW-USERS	alle aktiven Auftraggeber mit entsprechender Zusatzinformation ausgeben
STOP-DCN	SESDCN beenden

Tabelle 65: SESDCN-Administrationsanweisungen

8.9.4 E-Mail-Versand wichtiger Informationen der DBH-Session

Der SESAM/SQL-Systemverwalter kann wichtige Informationen der DBH-Session auch automatisch per E-Mail versenden lassen.

Detaillierte Informationen zum Aufbau der MAIL-Parameterdatei und zur E-Mail-Ausgabe des DBH (DBH-Optionen, Administrationsanweisungen und -kommandos) finden Sie im Handbuch „Datenbankbetrieb“

Voraussetzungen für die E-Mail-Ausgabe des DBH

Der Mail-Service des Softwareprodukts interNet Services muss zur Verfügung stehen.

Der eigentliche Versand der E-Mail erfolgt durch den DBH mit dem Mail-Sender (Schnittstelle SEND-MAIL) des Softwareprodukts interNet Services.

MAIL-Parameterdatei

Die MAIL-Parameterdatei ist eine BS2000-SAM-Datei, die der Anwender anlegt und beim Start dem DBH über den Linknamen SESMAIL zuweist, siehe Abschnitt „MAIL-Parameterdatei“ auf Seite 301.

In die MAIL-Parameterdatei tragen Sie den oder die Empfänger, den Absender sowie relevante Meldungsnummern für die E-Mail-Ausgaben des DBH ein.

Ohne MAIL-Parameterdatei können Sie die E-Mail-Ausgabe des DBH (zunächst) nicht nutzen.

Die MAIL-Parameterdatei kann in der laufenden DBH-Session geändert werden. Die (geänderten) Parameter werden aber erst durch die Eingabe der Administrationsanweisung MODIFY-MAIL-PARAMETERS wirksam. Die aktuellen MAIL-Parameter können Sie mit der Administrationsanweisung SHOW-MAIL-PARAMETERS ausgeben.

Optionen und Anweisungen zur Steuerung der E-Mail-Ausgabe des DBH

Den Umfang der E-Mail-Ausgabe des DBH stellen Sie mit den DBH-OPTIONEN MSG-OUTPUT und SERVICE-TASKS ein.
Mit den Administrationsanweisungen MODIFY-MSG-OUTPUT und MODIFY-SERVICE-TASKS bzw. den Administrationskommandos OPT,MSG-OUTPUT und OPT,SVT können Sie den Umfang der E-Mail-Ausgabe des DBH in der laufenden DBH-Session ändern.

Mit der Anweisung MODIFY-OUTPUT-MODE können Sie als Ausgabeziel für alle folgenden Administrationsanweisungen eine zusätzliche Ausgabe per E-Mail festlegen.

8.9.5 Auftragsprotokollierung auswerten mit SESCOSP

SESCOSP ist ein Dienstprogramm für die Ausgabe protokollierter Aufträge.

Wenn die Auftragsprotokollierung eingeschaltet ist, sammelt der DBH Daten in einer Auftragsprotokoll-Datei (CO-LOG-Datei). Das Dienstprogramm SESCOSP bereitet die gesammelten Auftragsdaten auf und gibt sie in wählbaren Listenformaten aus.

Mit der Auftragsprotokollierung kann der Systemverwalter den aufgezeichneten Zeitraum exakt verfolgen und beurteilen. Die Daten geben Auskunft darüber, welche Anweisungen für welche Programme ausgeführt wurden und welche Spaces und welche Datenbanken angesprochen wurden. So können SESCOSP-Auswertungen beispielsweise helfen, eine performancekritische Anweisung zu identifizieren.

Der Systemverwalter aktiviert die Auftragsprotokollierung mit der Administrationsanweisung SET-TUNING-TRACE (siehe Handbuch „Datenbankbetrieb“).
Die Auftragsprotokollierung belastet allerdings den DBH.
Die Bedienung von SESCOSP ist im Handbuch „Datenbankbetrieb“ beschrieben.

8.9.6 Betriebsdaten ausgeben mit SESMON

Der Performance-Monitor SESMON sammelt Daten über den laufenden Datenbankbetrieb, bereitet sie statistisch nach verschiedenen Gesichtspunkten auf und gibt sie als Bildschirmmasken, Druckerlisten, in Dateiform oder über den SESAM-MON-Subagenten an eine Management-Plattform aus. SESMON läuft als eigene Task und beeinflusst das Laufzeitverhalten des DBH nicht.

Der Systemverwalter erhält eine Vielzahl von Informationen über den DBH- oder den SESDCN-Betrieb, zum Beispiel:

- die aktuellen Werte der DBH- und DCN-Optionen
- Angaben zur Größe und Belegung der Container
- Informationen über den Fortschritt eines Wiederanlaufs der DBH-Session
- Angaben zu SQL-Anweisungen und SQL-Zugriffsplänen
- Angaben zu jeder aktiven Anwendung
- Anzahl und Auslastung der Service-Tasks
- Informationen über den Fortschritt eines RECOVER- oder REFRESH-Laufes
- Zugriffe auf Spaces und Cursor-Dateien
- Angaben zu den Warteschlangen
- Anzahl und Zustand momentan offener Transaktionen
- Belegung des Prefetch-Puffers

SESMON bietet dem Systemverwalter damit Unterlagen

- zur Performanceanalyse von SESAM/SQL
  Die Wirkung verschiedener Optionen auf das Zeitverhalten von DBH und SESDCN kann beobachtet werden.
- für einen optimalen Programm-Mix
  Die Auswirkungen parallel ablaufender Anwenderprogramme lassen sich feststellen.
- für Statistikzwecke
  Die Daten geben zum Beispiel Auskunft über die Häufigkeit einzelner Datenbankoperationen oder die Anzahl der Plattenzugriffe in einem bestimmten Zeitraum.

Zugang zum Performance-Monitor SESMON erhalten Sie auch aus dem World Wide Web, siehe Seite 38.

Die Bedienung von SESMON ist im Handbuch „Datenbankbetrieb“ beschrieben.

8.9.7 DA-LOG-Aufbereitung durch SEDI70

SEDI70 ist ein Dienstprogramm für die Ausgabe der Änderungen in Anwender-Spaces.

Wenn Logging eingeschaltet ist, sammelt der DBH Daten über Änderungen in den Anwenderdaten in einer Logging-Datei (DA-LOG-Datei). Das Dienstprogramm SEDI70 bereitet die gesammelten Änderungen auf und gibt sie in wählbaren Listenformaten aus.

Die Daten geben Auskunft darüber, welche SQL- oder CALL-DML-Anweisungen ausgeführt wurden und welche Spaces und welche Datenbanken angesprochen wurden. Außerdem ist in der DA-LOG-Datei die Erzeugung von SESAM-Sicherungsbeständen für einen Anwender-Space protokolliert.

Die Bedienung von SEDI70 ist im Handbuch „Datenbankbetrieb“ beschrieben.

9 Datenbank aufbauen, laden und warten

Dieses Kapitel beschreibt

- in welchen Schritten eine SESAM/SQL-Datenbank aufgebaut wird und wie der Aufbau geändert werden kann
- wie Anwenderdaten geladen und entladen werden
- welche Arbeiten im Zusammenhang mit der Wartung einer SESAM/SQL-Datenbank anfallen

Anweisungen und Werkzeuge für das Aufbauen und Warten einer Datenbank

Als komfortables Werkzeug für Aufbau und Wartung einer SESAM/SQL-Datenbank gibt es den Utility-Monitor. Mit dem Utility-Monitor kann der Datenbankverwalter alle Arbeiten im Zusammenhang mit Aufbau und Wartung einer SESAM/SQL-Datenbank menügeführt erledigen (siehe Handbuch „Utility-Monitor“).

Alternativ kann der Datenbankverwalter ESQL-Programme für Aufbau und Wartung einer SESAM/SQL-Datenbank erstellen. Hierfür stehen ihm folgende Anweisungsgruppen zur Verfügung:

- Utility-Anweisungen
- SQL-Anweisungen zur Verwaltung der Speicherstruktur
- SQL-Anweisung zur Verwaltung von Benutzereinträgen
- SQL-Anweisungen zur Schemadefinition und Schemaverwaltung

Ausführlich erklärt sind die SQL-Anweisungen im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“ und die Utility-Anweisungen im Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“.

9.1 Datenbank aufbauen und Aufbau ändern

Dieser Abschnitt beschreibt,

- wie der Grundaufbau einer SESAM/SQL-Datenbank durch Anlegen des Catalog-Space erstellt wird
- wie der Grundaufbau einer SESAM/SQL-Datenbank erweitert wird und wie der Aufbau einer SESAM/SQL-Datenbank geändert werden kann.

9.1.1 Grundaufbau einer SESAM/SQL-Datenbank erstellen

Der Grundaufbau einer SESAM/SQL-Datenbank wird mit der Utility-Anweisung CREATE CATALOG erstellt. Bei CREATE CATALOG legt SESAM/SQL den Catalog-Space der Datenbank an.

Die folgende Tabelle gibt einen Überblick über die erforderlichen Tätigkeiten und ordnet den Tätigkeiten die entsprechenden Utility-Anweisungen zu. Die rechte Spalte zeigt die entsprechende Hauptfunktion des Utility-Monitors.

Tätigkeit	**Anweisung und Funktion**	**Hauptfunktion im Utility-Monitor**
Catalog-Space der Datenbank anlegen auf Seite 341	CREATE CATALOG Festlegen von – BS2000-Kennwort für die Datenbank – DB-Kennung – Zeichensatz (Code-Tabelle) – Eigenschaften des Catalog-Space – Storage Group für den Catalog-Space – Storage Group für CAT-REC- und 1. CAT-LOG-Datei – Universeller Benutzer	Hauptfunktion CREATE CATALOG (CRC) mit Folgemasken
Catalog-Space sichern auf Seite 342	COPY Datenbank (Catalog-Space) sichern; nicht unbedingt erforderlich	

Tabelle 66: Utility-Anweisungen und Hauptfunktion des Utility-Monitors für den Grundaufbau einer Datenbank

9.1.1.1 Catalog-Space der Datenbank anlegen

Den Catalog-Space der Datenbank legt ein dazu autorisierter Benutzer mit der Utility-Anweisung CREATE CATALOG an.

Die Berechtigung zur Ausführung der Utility-Anweisung CREATE CATALOG für eine bestimmte System-Benutzerkennung vergibt der Systemverwalter, indem er diese System-Benutzerkennung dem zuständigen SESAM/SQL-DBH bekannt gibt mit der DBH-Option ADMINISTRATOR bzw. mit der Administrationsanweisung MODIFY-ADMINISTRATION (siehe Handbuch „Datenbankbetrieb“).

Der Catalog-Space kann in der DBH-Kennung oder in einer DB-Kennung angelegt werden, siehe Abschnitt „Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen“ auf Seite 308.

Der Catalog-Space kann auf Pubsets mit „großen Dateien“ bis zu 4 TByte groß werden, siehe Seite 351. Sonst kann er bis zu 64 GByte groß werden.

Der Catalog-Space nimmt die Metadaten der Datenbank auf. SESAM/SQL initialisiert die leeren Metadaten, indem es die bei CREATE CATALOG gemachten Angaben sowie weitere Verwaltungsinformationen (z.B. die Views des Informationsschemas) einträgt.
Aus Sicht des Betriebssystems ist der Catalog-Space eine gewöhnliche BS2000-Datei. Eine Initialisierung der Datenbank mit CREATE CATALOG ist in jedem Fall erforderlich. Der BS2000-Dateiname des Catalog-Space lautet:

:*catid*:*benutzerkennung.catalog.*CATALOG

BS2000-Benutzerkennung für die Datenbank

Alle Dateien einer SESAM/SQL-Datenbank außer Logging- und Fehlerdateien müssen in derselben BS2000-Benutzerkennung katalogisiert sein. SESAM/SQL trägt die Benutzerkennung für die Datenbank in das SQL-Datenbankverzeichnis des DBH ein (siehe Handbuch „Datenbankbetrieb“).

Bei Datenbanken, die nicht in der Benutzerkennung des DBH liegen, ist zu beachten:

- Erstellen von SESAM-Sicherungsbeständen auf Magnetbandkassette ist nur mit HSMS möglich. Dazu muss auf der DB-Kennung die DBH-Kennung als Miteigentümer an den DB-Dateien definiert werden. Reparatur oder Zurücksetzen auf der Basis eines SESAM-Sicherungsbestandes auf Magnetbandkassette ist ebenfalls nur mit HSMS möglich.
- SESAM/SQL kann die SQL-Anweisung CREATE SPACE nur dann ausführen, wenn die Vorbereitungen für das Anlegen des Catalog-Space getroffen worden sind, siehe Abschnitt „Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen“ auf Seite 308.

- Sollen Logging-Dateien (CAT-LOG-Dateien, DA-LOG-Dateien) in der DB-Kennung liegen, so müssen die Vorbereitungen für das Anlegen der Logging-Dateien getroffen worden sein, siehe Abschnitt „Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen“ auf Seite 308.
  Können Logging-Dateien nicht auf der DB-Kennung angelegt werden, legt SESAM/SQL sie in der DBH-Kennung an.
  Bei der Reparatur sucht SESAM/SQL die Logging-Dateien zuerst in der DB-Kennung und anschließend in der DBH-Kennung.

9.1.1.2 Catalog-Space ändern

Der universelle Benutzer kann den Catalog-Space mit der SQL-Anweisung ALTER SPACE "CATALOG" (CATALOG in Anführungszeichen) ändern.

Mit ALTER SPACE kann die Freiplatzreservierung verändert und eine neue Storage Group für den Catalog-Space angegeben werden. Wird die Storage Group geändert, muss der universelle Benutzer das Sonder-Privileg USAGE für die neue Storage Group besitzen.

Die Änderungen werden bei der nächsten Reorganisation des Catalog-Space wirksam.

9.1.1.3 Catalog-Space sichern

Nach dem Grundaufbau der Datenbank sollte der Datenbankverwalter mit der Utility-Anweisung COPY einen SESAM-Sicherungsbestand des Catalog-Space erstellen. Dazu benötigt der Datenbankverwalter das Privileg UTILITY. Wie SESAM-Sicherungsbestände mit der Utility-Anweisung COPY erstellt werden, ist im Abschnitt „SESAM-Sicherungsbestand erstellen mit COPY“ auf Seite 368 näher beschrieben.

9.1.2 Grundaufbau einer Datenbank erweitern und Aufbau ändern

Im Grundaufbau enthält eine SESAM/SQL-Datenbank nur den Catalog-Space und, sofern Logging vereinbart ist, die CAT-REC-Datei sowie die erste CAT-LOG-Datei. Der Grundaufbau kann durch folgende Maßnahmen zu einer einsatzfähigen SESAM/SQL-Datenbank erweitert werden, die dann auch Anwenderdaten aufnehmen kann:

- Systemzugänge erzeugen
- Sonder-Privilegien vergeben
- Storage Groups für Anwender-Spaces definieren
- Ersten Mediensatz für DA-LOG- und PBI-Dateien bzw. die HSMS-Arbeitsdatei in die Medientabelle aufnehmen
- Anwender-Spaces anlegen
- Schemata erstellen

Der Aufbau einer einsatzfähigen SESAM/SQL-Datenbank lässt sich wie folgt ändern und so den jeweils aktuellen Erfordernissen anpassen:

- Systemzugänge erzeugen und löschen
- Sonder-Privilegien vergeben und entziehen
- Storage Groups für Anwender-Spaces definieren, ändern, löschen
- Medientabelle pflegen, d.h. Mediensätze neu aufnehmen oder löschen, Beschreibung der Dateieigenschaften in der Medientabelle ändern
- Anwender-Spaces anlegen, ändern, löschen
- Schemata erstellen, ändern, löschen
- Eigenschaften der Datenbank ändern

Die genannten Maßnahmen sind in den folgenden Abschnitten näher erläutert.

Die folgende Tabelle gibt einen Überblick über die erforderlichen Tätigkeiten und ordnet den Tätigkeiten die entsprechende SQL- bzw. Utility-Anweisung zu. Die rechte Spalte zeigt, mit welchen Hauptfunktionen des Utility-Monitors der Datenbankverwalter die jeweilige Aufgabe erledigen kann.

Tätigkeit	**Anweisung und Funktion**	**Hauptfunktion im Utility-Monitor**
Systemzugänge erzeugen und löschen	CREATE USER Berechtigungsschlüssel (für SQL-Benutzer) vergeben CREATE SYSTEM_USER Systemzugang erzeugen DROP USER Berechtigungsschlüssel und zugehörige Systemzugänge löschen DROP SYSTEM_USER Systemzugang löschen	Hauptfunktion CREATE CATALOG (CRC) mit Folgemasken sowie Hauptfunktion ALTER CATALOG (ALC) mit Folgemasken
Sonder-Privilegien vergeben und entziehen	GRANT Sonder-Privileg vergeben REVOKE Sonder-Privileg entziehen	
Storage Groups definieren, ändern, löschen	CREATE STOGROUP Storage Group definieren ALTER STOGROUP Storage Group ändern DROP STOGROUP Storage Group löschen	Hauptfunktion SSL mit Folgemasken
Ersten Mediensatz für DA-LOG- und PBI-Dateien in die Medientabelle aufnehmen	CREATE MEDIA DESCRIPTION Jeweils ersten Mediensatz für die datenbankspezifischen Dateien DA-LOG und PBI bzw. die HSMS-Arbeitsdatei in die Medientabelle aufnehmen	Hauptfunktion CREATE CATALOG (CRC) mit Folgemasken
Medientabelle pflegen	ALTER MEDIA DESCRIPTION Mediensätze neu aufnehmen oder löschen; Beschreibung der Dateieigenschaften ändern DROP MEDIA DESCRIPTION Alle Mediensätze für eine bestimmte Dateiart löschen	Hauptfunktion ALTER CATALOG (ALC) mit Folgemasken

Tabelle 67: SQL-/Utility-Anweisungen und Hauptfunktionen des Utility-Monitors zum Erweitern und Ändern des Aufbaus einer Datenbank (Teil 1 von 2)

Tätigkeit	**Anweisung und Funktion**	**Hauptfunktion im Utility-Monitor**
Anwender-Spaces anlegen, ändern, löschen	CREATE SPACE Anwender-Space anlegen ALTER SPACE Anwender-Space ändern DROP SPACE Anwender-Space löschen	Hauptfunktion SSL mit Folgemasken
Schemata erstellen, ändern, löschen	CREATE SCHEMA Schema erstellen Schema ändern DROP SCHEMA Schema löschen	Hauptfunktion CREATE SCHEMA (CRS) mit Folgemasken Hauptfunktion ALTER SCHEMA (ALS) mit Folgemasken
Eigenschaften der Datenbank ändern	ALTER CATALOG Eigenschaften ändern	Hauptfunktion ALTER CATALOG (ALC) mit Folgemasken

Tabelle 67: SQL-/Utility-Anweisungen und Hauptfunktionen des Utility-Monitors zum Erweitern und Ändern des Aufbaus einer Datenbank (Teil 2 von 2)

9.1.2.1 Systemzugänge erzeugen und löschen

Um mit SQL eine SESAM/SQL-Datenbank bearbeiten zu können, benötigt ein BS2000-Benutzer einen Berechtigungsschlüssel eines SQL-Benutzers. Für jeden Berechtigungsschlüssel muss mindestens ein entsprechender Systemzugang in den Metadaten der Datenbank hinterlegt sein (siehe Abschnitt „Systemzugang“ auf Seite 172).

Systemzugang erzeugen

Der Datenbankverwalter vergibt den Berechtigungsschlüssel für einen SQL-Benutzer mit der SQL-Anweisung CREATE USER.

Mit der SQL-Anweisung CREATE SYSTEM_USER definiert der Datenbankverwalter einen Systemzugang.

Bevor mit CREATE SYSTEM_USER ein Systemzugang erzeugt werden kann, muss der zugehörige Berechtigungsschlüssel des SQL-Benutzers mit CREATE USER vergeben worden sein. Pro Berechtigungsschlüssel kann der Datenbankverwalter mehrere Systemzugänge erzeugen.
Ein SQL-Benutzer der Datenbank wird repräsentiert durch alle Systemzugänge in den Metadaten der Datenbank, die den Berechtigungsschlüssel dieses SQL-Benutzers enthalten.

Systemzugang löschen

Der Datenbankverwalter löscht den Berechtigungsschlüssel eines SQL-Benutzers mit der SQL-Anweisung DROP USER.

DROP USER löscht einen Berechtigungsschlüssel und alle zugehörigen Systemzugänge. Ein Berechtigungsschlüssel kann nicht gelöscht werden, wenn er Eigentümer von Schemata, Anwender-Spaces oder Storage Groups ist oder wenn er Grantor eines Privilegs ist. Der Berechtigungsschlüssel des universellen Benutzers kann nicht gelöscht werden.

Einzelne Systemzugänge für einen SQL-Benutzer löscht der Datenbankverwalter mit der SQL-Anweisung DROP SYSTEM_USER.

Ein Systemzugang kann nicht gelöscht werden, wenn dieser Systemzugang der einzige Systemzugang für den universellen Benutzer ist.

9.1.2.2 Sonder-Privilegien vergeben und entziehen

Unabhängig davon, wieviele Systemzugänge bereits eingerichtet sind, ist zunächst nur der universelle Benutzer berechtigt, die neue Datenbank mit SQL- und Utility-Anweisungen zu bearbeiten. Damit weitere SQL-Benutzer Teilaufgaben der Datenbankverwaltung übernehmen können, müssen diese selbst berechtigt sein, beispielsweise Utility-Anweisungen abzusetzen. Ferner müssen einzelne Benutzer die Berechtigung erhalten, Schemata zu definieren (siehe Abschnitt „Schemata erstellen, ändern und löschen“ auf Seite 352). Zu diesem Zweck kann der universelle Benutzer Sonder-Privilegien für folgende Tätigkeiten mit der SQL-Anweisung GRANT erteilen:

- Einrichten und Löschen von SQL-Benutzern (CREATE USER)
- Erstellen eines Schemas (CREATE SCHEMA)
- Ausführen der Utility-Anweisungen (UTILITY)
- Festlegen der Speichergeräte für die Spaces (CREATE STOGROUP)
- Anlegen von Anwender-Spaces auf einer bestimmten Storage Group (USAGE ON STOGROUP)
- Alle Sonder-Privilegien (ALL SPECIAL PRIVILEGES)

Der universelle Benutzer hat die Berechtigung zur Vergabe von Sonder-Privilegien mit Ausnahme von USAGE ON STOGROUP. Der Eigentümer dieser Storage Group hat die Berechtigung zur Vergabe des Sonderprivilegs USAGE ON STOGROUP für eine bestimmte Storage Group.

Jedes mit der SQL-Anweisung GRANT erteilte Sonder-Privileg kann der Grantor dieses Privilegs dem Grantee mit der SQL-Anweisung REVOKE wieder entziehen.

Näheres zu Sonder-Privilegien sowie zur Vergabe von Privilegien allgemein ist im Abschnitt „Zugriffsschutz in SQL durch Privilegien“ auf Seite 176 beschrieben.

9.1.2.3 Storage Groups definieren, ändern und löschen

Eine Storage Group definiert Speichermedien (Privatplatten mit zugehörigem Gerätetyp oder PUBLIC-Platten) und eine BS2000-Katalogkennung, die für das Anlegen von Anwender-Spaces und SESAM-Sicherungsbeständen benötigt werden.

Storage Group definieren

Eine Storage Group definiert der Datenbankverwalter mit der SQL-Anweisung CREATE STOGROUP.

Storage Group ändern

Der Datenbankverwalter kann die Definition einer Storage Group mit der SQL-Anweisung ALTER STOGROUP ändern.

Die Anweisung ALTER STOGROUP ändert nur die Definition der Storage Group. Bereits bestehende Anwender-Spaces, die die Platten der Storage Group verwenden, sind nicht betroffen.

Aus der Definition entfernte Platten werden nicht mehr für neue Speicherzuweisungen an die Anwender-Spaces verwendet. Platten können explizit durch die Klausel DROP VOLUME gelöscht werden oder implizit durch den Wechsel von gemeinschaftlichen Platten (PUBLIC) auf Privatplatten bzw. umgekehrt.

Storage Group löschen

Eine bereits vorhandene Storage Group kann der Datenbankverwalter mit der SQL-Anweisung DROP STOGROUP löschen.

DROP STOGROUP wird abgewiesen, solange auf den durch die betreffende Storage Group definierten Speichermedien noch ein Anwender-Space liegt.

9.1.2.4 Ersten Mediensatz für DA-LOG- und PBI-Dateien in die Medientabelle aufnehmen

Angaben zu Speichermedien und Eigenschaften für die datenbank-spezifischen Dateien (CAT-REC-Datei, CAT-LOG-Dateien, DA-LOG-Dateien und PBI-Datei bzw. die HSMS-Arbeitsdatei) sind in der Medientabelle als sogenannte Mediensätze hinterlegt (siehe Seite 212). Der jeweils erste Mediensatz für die CAT-REC-Datei und die CAT-LOG-Dateien wird bereits bei Ausführung der Utility-Anweisung CREATE CATALOG erzeugt (siehe Abschnitt „Catalog-Space der Datenbank anlegen“ auf Seite 341).

Den jeweils ersten Mediensatz für DA-LOG-Dateien und die PBI-Datei bzw. die HSMS-Arbeitsdatei erzeugt der Datenbankverwalter mit der Utility-Anweisung CREATE MEDIA DESCRIPTION.

Die Ausführung der Anweisung CREATE MEDIA DESCRIPTION für eine bestimmte Dateiart wird abgelehnt, wenn es für die betreffende Dateiart bereits einen Eintrag in der Medientabelle gibt.

Für die Pflege der Medientabelle, d.h. um weitere Mediensätze aufzunehmen oder um Mediensätze zu löschen, gibt es die Utility-Anweisungen ALTER MEDIA DESCRIPTION und DROP MEDIA DESCRIPTION.

9.1.2.5 Medientabelle pflegen

Mediensätze neu aufnehmen oder löschen,
Beschreibung der Dateieigenschaften ändern

Mit der Utility-Anweisung ALTER MEDIA DESCRIPTION kann der Datenbankverwalter für eine bestimmte datenbank-spezifische Dateiart die Beschreibung der Dateieigenschaften in der Medientabelle ändern. Außerdem lassen sich mit ALTER MEDIA DESCRIPTION Mediensätze in die Medientabelle neu aufnehmen oder Mediensätze löschen.

Die mit ALTER MEDIA DESCRIPTION vorgenommenen Änderungen betreffen nur neu anzulegende Dateien der angegebenen Dateiart. Auf bereits bestehende Dateien hat ALTER MEDIA DESCRIPTION keine Auswirkungen.

Alle Mediensätze für eine bestimmte Dateiart löschen

Mit der Utility-Anweisung DROP MEDIA DESCRIPTION kann der Datenbankverwalter alle Mediensätze für eine bestimmte datenbank-spezifische Dateiart aus der Medientabelle löschen.

Auf Dateien, deren Eigenschaften in den zu löschenden Mediensätzen beschrieben werden, hat DROP MEDIA DESCRIPTION keine Auswirkungen, d.h. bestehende Dateien werden nicht gelöscht.

9.1.2.6 Anwender-Spaces anlegen, ändern und löschen

Anwender-Spaces dienen zur Aufnahme der Anwenderdaten einer SESAM/SQL-Datenbank, d.h. zur Aufnahme von Anwendertabellen und darauf definierten Indizes. Aus Sicht des Betriebssystems sind Anwender-Spaces gewöhnliche BS2000-Dateien.

Anwender-Space anlegen

Liegt die Datenbank, zu der ein Anwender-Space gehören soll, in einer DB-Kennung, muss der Datenbankverwalter Vorbereitungen für das Anlegen des neuen Anwender-Space in dieser BS2000-Benutzerkennung treffen, siehe Abschnitt „Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen“ auf Seite 308.

Einen Anwender-Space legt der Datenbankverwalter mit der SQL-Anweisung CREATE SPACE an. Maximal können 999 Anwender-Spaces pro Datenbank angelegt werden.
Ein Anwender-Space kann auf Pubsets mit „großen Dateien“ bis zu 4 TByte groß werden, siehe Seite 351. Sonst kann er bis zu 64 GByte groß werden.

Anwender-Space ändern

Der Eigentümer eines Anwender-Space kann diesen Anwender-Space mit der SQL-Anweisung ALTER SPACE ändern.

Mit ALTER SPACE kann die Freiplatzreservierung verändert und das DA-Logging ausgeschaltet werden. Außerdem kann in der USING-Klausel eine neue Storage Group für den Anwender-Space angegeben werden. Der Eigentümer des Anwender-Space benötigt dazu das Sonderprivileg USAGE ON STOGROUP für die neue Storage Group (siehe Abschnitt „Sonder-Privilegien“ auf Seite 176).
Eine Änderung der Freiplatzreservierung wird bei der nächsten Reorganisation des Anwender-Space wirksam. Vorzeitig wirksam wird die Änderung

- bei IMPORT TABLE, wenn Tabellen einschließlich ihrer Anwenderdaten importiert werden,
- bei LOAD OFFLINE, sowohl wenn Daten in eine leere Tabelle zugeladen werden als auch wenn neue Daten an bereits vorhandene Daten angefügt werden.

Eine Änderung der Storage Group wird erst bei der nächsten Reorganisation des Anwender-Space wirksam.

Anwender-Space löschen

Der Eigentümer eines Anwender-Space kann diesen Anwender-Space mit der SQL-Anweisung DROP SPACE löschen.

9.1.2.7 Freiplatzreservierung für Catalog-Space und Anwender-Spaces

Über die Freiplatzreservierung kann eingestellt werden, bis zu welchem Grad die Blöcke des Catalog-Space oder eines Anwender-Space mit Daten gefüllt werden sollen. Ein günstig gewählter Wert für die Freiplatzreservierung verbessert die Performance bei der Datenänderung.

SESAM/SQL berücksichtigt die Freiplatzreservierung beim Zuladen von Anwenderdaten, beim Importieren einer Tabelle mit ihren Anwenderdaten, beim Ändern von Partitionsgrenzen , beim Hinzufügen, Ändern oder Löschen von Spalten und beim Vertauschen von Spaltenwerten

9.1.2.8 Spaces mit einer Größe von mehr als 64 GBytes

Ein Anwender-Space (und auch der Catalog-Space), der mit SESAM/SQL auf einem Pubset, das „große Dateien“ unterstützt, angelegt wird (mit CREATE SPACE, CREATE CATALOG oder RECOVER TO), kann bis zu 4 TByte groß werden („großer Space“). Sonst kann er bis zu 64 GByte groß werden.

Die maximale Spacegröße kann aus dem SYS_INFO_SCHEMA ermittelt werden.
Der View SYS_SPACE_PROPERTIES (im Utility-Monitor über die Funktion SNF) zeigt dazu bei der Spaceeigenschaft MAX_POSSIBLE_PAGE die größte mögliche Seitennummer an:

- 1.073.741.822 (X'3FFFFFFE') für Spaces bis 4 TByte
- 16.777.214 (X'00FFFFFE') für Spaces bis 64 GByte

Spaces aus früheren SESAM/SQL-Versionen können mit der Utility-Anweisung REORG SPACE in „große Spaces“ migriert werden sofern die Spaces auf einem Pubset liegen, das große Dateien unterstützt. Nach der Migration sollten Sie einen SESAM-Sicherungsbestand erstellen.

Wenn nach der Migration kein SESAM-Sicherungsbestand erstellt wird, der Space im laufenden Betrieb die 64GByte Grenze überschreitet und danach eine Recovery des Spaces notwendig wird, dann muss eine Sicherung des „alten“ Space verwendet werden. Dabei kommt es zum Abbruch der Recovery, weil während des Nachfahrens der Logging Dateien die 64GByte Grenze überschritten wird.

Dieses Problem kann man umgehen, indem man zuerst aus der Sicherung des „alten“ Space ein Replikat erzeugt. Das Replikat wird dann mit REORG SPACE reorganisiert und anschließend wird die Recovery mit dem Replikat durchgeführt
(RECOVER … USING REPLICATION).

9.1.2.9 Schemata erstellen, ändern und löschen

Schemata enthalten Metadaten, die den formalen Aufbau von Basistabellen, Indizes, Views und Routinen beschreiben, sowie Metadaten, die die Privilegien beschreiben.

Schema erstellen

Mit der SQL-Anweisung GRANT vergibt der Datenbankverwalter das Sonder-Privileg CREATE SCHEMA an einen SQL-Benutzer. Er ist dann berechtigt, ein Schema und damit Objekte für dieses Schema zu erstellen (siehe Abschnitt „Sonder-Privilegien“ auf Seite 176). Der so berechtigte Benutzer erstellt ein Schema mit der SQL-Anweisung CREATE SCHEMA.

Der CREATE SCHEMA-Berechtigte kann bereits beim Erstellen des Schemas SQL-Objekte (Basistabellen, Views, Indizes, Routinen) definieren sowie Tabellen- und EXECUTE-Privilegien vergeben, indem er innerhalb von CREATE SCHEMA die SQL-Anweisungen CREATE TABLE, CREATE VIEW, CREATE INDEX, GRANT, CREATE PROCEDURE und CREATE FUNCTION angibt.

Mit der SQL-Anweisung GRANT kann der Schema-Eigentümer Tabellen- und EXECUTE-Privilegien für Objekte seines Schemas an andere Benutzer weitergeben (siehe Abschnitt „Tabellen-Privilegien“ auf Seite 178).

Schema ändern

Der Eigentümer des Schemas kann es an die aktuellen Erfordernisse anpassen:

- Basistabellen erzeugen, ändern oder löschen.
  Basistabellen erzeugt der Schema-Eigentümer mit der SQL-Anweisung CREATE TABLE, bestehende Basistabellen ändert bzw. löscht er mit den SQL-Anweisungen ALTER TABLE bzw. DROP TABLE. Näheres zum Erzeugen, Ändern und Löschen von Basistabellen ist im Abschnitt „Basistabelle“ auf Seite 85 beschrieben.
- Views erzeugen oder löschen.
  Views erzeugt der Schema-Eigentümer mit der SQL-Anweisung CREATE VIEW, bestehende Views löscht er mit der SQL-Anweisung DROP VIEW. Näheres zum Erzeugen und Löschen von Views ist im Abschnitt „View“ auf Seite 90 beschrieben.
- Indizes erzeugen oder löschen.
  Indizes erzeugt der Schema-Eigentümer mit der SQL-Anweisung CREATE INDEX, bestehende Indizes löscht er mit der SQL-Anweisung DROP INDEX. Näheres zum Erzeugen und Löschen von Indizes ist im Abschnitt „Index“ auf Seite 96 beschrieben.

- Routinen erzeugen oder löschen.
  Routinen erzeugt der Schema-Eigentümer mit den SQL-Anweisungen CREATE PROCEDURE und CREATE FUNCTION, bestehende Routinen löscht er mit den Anweisungen DROP PROCEDURE und DROP FUNCTION. Näheres zum Erzeugen und Löschen von Prozeduren ist im Abschnitt „Routine“ auf Seite 98 beschrieben.
- Tabellen- und EXECUTE-Privilegien erteilen oder entziehen.
  Privilegien für Tabellen und Routinen seines Schemas erteilt der Schema-Eigentümer mit der SQL-Anweisung GRANT. Entzogen werden Privilegien mit der SQL-Anweisung REVOKE. Näheres zum Erteilen und Entziehen von Privilegien ist im Abschnitt „Zugriffsschutz in SQL durch Privilegien“ auf Seite 176 beschrieben.

Schema löschen

Mit der SQL-Anweisung DROP SCHEMA kann der Schema-Eigentümer ein Schema löschen.

9.1.2.10 Eigenschaften der Datenbank ändern

Mit der SQL-Anweisung ALTER CATALOG kann der codierte Zeichensatz (Coded Character Set, CCS) der Datenbank geändert werden.

Wenn `CODE_TABLE _NONE_` angegeben wird, dann wird für die Datenbank kein codierter Zeichensatz verwendet. Dies entspricht dem Verhalten von SESAM/SQL vor V5.0.

9.2 Basistabellen exportieren und importieren

Mit EXPORT TABLE wird eine Basistabelle einer SESAM/SQL-Datenbank in eine BS2000-Datei exportiert. Diese BS2000-Datei wird Export-Datei genannt. Die Export-Datei wird mit einem Coded Character Set Name (CCSN) gemäß dem Zeichensatz der Datenbank angelegt. Beim Importieren wird der CCSN der Export-Datei gegen den codierten Zeichensatz der Datenbank geprüft.

In der Export-Datei werden Metadaten der Tabelle abgelegt. Wahlweise können auch Anwenderdaten exportiert werden. Aus dieser Export-Datei wird mit IMPORT TABLE eine neue Basistabelle aufgebaut. Die neue Basistabelle kann auch eine partitionierte Tabelle sein.

Im Einzelnen können Sie das Anweisungspaar EXPORT/IMPORT TABLE für folgende Aufgaben nutzen:

- Kopieren einer Basistabelle innerhalb einer Datenbank oder von einer Datenbank in eine andere:
  Mit Hilfe einer Export-Datei kann eine Basistabelle auf einfache Weise kopiert werden, da die Struktur der Zieltabelle über die Metadaten in der Export-Datei abgelegt ist.
  Im Vergleich zu dem Anweisungspaar LOAD/UNLOAD sind deutlich weniger Arbeitsschritte nötig, weil bei LOAD die Tabellenstruktur neu eingerichtet werden muss.
  Über eine Suchbedingung in der EXPORT TABLE-Anweisung können Sie auswählen, welche Anwenderdaten in die neue Tabelle übertragen werden sollen.
  Anders als die bei LOAD/UNLOAD verwendeten Dateien kann die Export-Datei jedoch nicht händisch bearbeitet werden.
  Die Export-Datei ist deutlich kleiner als eine mit UNLOAD erzeugte Datei (siehe Abschnitt „Anwenderdaten entladen mit UNLOAD“ auf Seite 361).
- Verlagern einer Basistabelle von einem Space auf einen anderen:
  Dazu wird die Basistabelle exportiert und anschließend mit DROP TABLE gelöscht. Importiert werden kann sie auf einem beliebigen anderen Space mit IMPORT TABLE. Views, Referenzbedingungen oder Zugriffsrechte auf diese Tabelle müssen gegebenenfalls angepasst werden.
- Umwandeln einer Basistabelle in eine partitionierte Tabelle oder Modifizieren von Anzahl und Grenzen der Partitionen einer partitionierten Tabelle:
  Dazu wird die Basistabelle exportiert und anschließend ggf. mit DROP TABLE gelöscht. Importiert werden kann sie als neue, partitionierte Tabelle mit gleichem oder unterschiedlichem Namen unter Angabe der (veränderten) Partitionsstruktur mit IMPORT TABLE.
  Dafür steht auch die komfortable Utility-Anweisung ALTER PARTITIONING FOR TABLE zur Verfügung, siehe Abschnitt „Partitionierung einer Basistabelle ändern“ auf Seite 379.

- Restrukturierung einer Basistabelle:
  Eine Basistabelle lässt sich in mehrere neue Tabellen aufteilen.
  Dazu wird sie exportiert und anschließend mehrfach unter verschiedenen Namen importiert. Diese zunächst identischen neuen Basistabellen können mit ALTER TABLE verändert werden, bis die gewünschte Aufteilung erreicht ist.
- Neuvergabe der Satznummern einer Basistabelle:
  Bei Basistabellen mit Primärschlüsseln können die Satznummern neu vergeben werden.
  Dazu wird eine Tabelle exportiert, gelöscht und auf dem selben Space wieder importiert unter Angabe von NEW ROW_IDS.
- Reorganisation einer Tabelle:
  Die Anweisung IMPORT TABLE kann zu einer Reorganisation auf Tabellenebene genutzt werden, da bei IMPORT TABLE der Blockfüllgrad des Space berücksichtigt wird.
  Dazu wird eine Tabelle exportiert, gelöscht und auf dem selben Space wieder importiert.

  Die Reorganisation einer Tabelle kann auch mit REORG ONLINE TABLE durchgeführt werden.
- Archivieren einer einzelnen Basistabelle:
  Mit EXPORT TABLE erzeugte Export-Dateien können als Sicherungskopien einzelner Tabellen verwendet werden. Eine Speicherung ist auch auf Magnetbandkassette möglich. Im Gegensatz zu Sicherungen mit der COPY-Anweisung wird hier nicht das Granulat Space, sondern das feinere Granulat Tabelle verwendet. Die Sicherung erfolgt in kompakter Form, da weniger Verwaltungsinformationen gespeichert werden müssen. Eine RECOVER-Funktion auf Basis dieser Sicherung existiert jedoch nicht. EXPORT-Dateien werden ausschließlich von IMPORT TABLE verarbeitet.

Eine Export-Datei, die eine Tabelle mit Spalten vom Datentyp NATIONAL CHARACTER (VARYING) enthält, kann nicht mehr mit SESAM/SQL bis V4.0 importiert werden.

9.2.1 Basistabellen exportieren mit EXPORT TABLE

Mit der Utility-Anweisung EXPORT TABLE kann der Datenbankverwalter eine Basistabelle, die auch partitioniert sein kann, aus einer SESAM/SQL-Datenbank in eine SAM-Datei mit variabler Satzlänge schreiben.

Es entsteht eine sogenannte Export-Datei, die nur von IMPORT TABLE interpretiert wird. Die exportierte Tabelle und eine aus der Export-Datei generierte neue Tabelle haben den gleichen Aufbau.

Die Export-Datei wird mit einem Coded Character Set Name (CCSN) gemäß dem Zeichensatz der Datenbank angelegt.

In der Export-Datei werden folgende Metadaten der Datenbank abgelegt:

- die Metadaten der Basistabelle und aller dazugehörigen Spalten
- die Metadaten für alle Indizes auf dieser Basistabelle
- die Metadaten aller Integritätsbedingungen mit Ausnahme von Referenzbedingungen, die auf dieser Basistabelle definiert sind

Über eine WHERE-Klausel können Sie die Menge der Anwenderdaten einschränken, die in die Export-Datei übertragen werden sollen. Die Export-Datei kann nicht händisch bearbeitet werden.

9.2.2 Basistabellen importieren mit IMPORT TABLE

Mit der Utility-Anweisung IMPORT TABLE kann der Datenbankverwalter eine neue Basistabelle aus den Informationen der Export-Datei in ein Schema einer SESAM/SQL-Datenbank laden. Der CCSN der Export-Datei muss mit dem codierten Zeichensatz der Datenbank (CODE_TABLE) übereinstimmen. Wenn für die Datenbank kein codierter Zeichensatz verwendet wird, dann wird auch der CCSN der Export-Datei ignoriert. Die neue Basistabelle kann auch eine partitionierte Tabelle sein.

Aus den Metadaten der Export-Datei wird in der angegebenen Datenbank eine Basistabelle, die partitioniert sein kann, erzeugt:

- Die Tabelle wird neu angelegt.
- Der Space bzw. die Beschreibung der Partitionen wird der Anweisung entnommen. Für partitionierte Tabellen müssen die Spaces bereits erzeugt sein.
- Der Tabellentyp wird von der exportierten Tabelle übernommen.
- Alle Spalten der exportierten Tabelle werden übernommen.
  Datentyp, Default-Eigenschaften und Reihenfolge der Spalten bleiben erhalten.
- Existiert ein Primärschlüssel, wird dieser übernommen.
  Für partitionierte Tabellen muss ein Primärschlüssel vorhanden sein.

Sie können wählen, ob die Anwenderdaten übernommen werden sollen, die in der Export-Datei abgelegt worden sind. Wenn Sie die Anwenderdaten importieren, können Sie auch dazugehörige Indizes bzw. Integritätsbedingungen aus der Export-Datei in die neue Tabelle einfügen.

Unterbrechung des Logging bei IMPORT TABLE

Liegt eine Basistabelle auf einem Anwender-Space, für den Logging vereinbart ist, protokolliert SESAM/SQL alle DML- bzw. DDL-Anweisungen, die den Tabelleninhalt bzw. die Tabellendefinition ändern, auf DA-LOG- bzw. CAT-LOG-Dateien.
Wenn mit IMPORT TABLE Daten in diese Tabelle geladen werden, wird das Logging unterbrochen. Werden auch Integritätsbedingungen mit IMPORT TABLE in die Tabelle geladen, ist garantiert, dass die Integritätsbedingungen erfüllt sind.
Falls das Logging unterbrochen wurde, muss der Datenbankverwalter nach dem IMPORT TABLE einen Sicherungsbestand des Anwender-Space erstellen, auf dem die Tabelle liegt (siehe Seite 368). Andernfalls verbleibt der Anwender-Space im Zustand „copy pending“ und kann nur mit Utility-Anweisungen oder Retrival-Anweisungen bearbeitet werden (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“).

9.3 Anwenderdaten zuladen und entladen

LOAD / UNLOAD bietet folgende Möglichkeiten:

- Anwenderdaten aus einer Eingabedatei in eine Basistabelle zu laden
- Anwenderdaten aus einer Basistabelle oder einem View in eine Ausgabedatei zu entladen
- Anwenderdaten von einer Basistabelle oder einem View in eine andere Basistabelle zu transferieren

Anwenderdaten offline oder online zu- oder entladen

Bei der Ausführung der beiden Anweisungen können Sie zwischen den Betriebsarten OFFLINE und ONLINE wählen. Die wesentlichen Unterschiede betreffen die Performance, das Sperrverhalten sowie die Fehlerbehandlung.

Die Betriebsart OFFLINE verhält sich wie der herkömmliche LOAD. Die Verarbeitung findet in einer Service-Task statt. Der Anwender-Space wird exklusiv gegen Ändern gesperrt.

In der Betriebsart ONLINE wird eine bessere Performance bei der Bearbeitung kleiner Dateien erreicht. Der Anwender-Space wird nicht exklusiv gegen Ändern gesperrt.

Detaillierte Informationen zu OFFLINE/ONLINE finden Sie auf bei der Beschreibung der Utility-Anweisungen im Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“.

9.3.1 Anwenderdaten zuladen mit LOAD

Mit der Utility-Anweisung LOAD kann der Datenbankverwalter Anwenderdaten aus einer Eingabedatei in eine Basistabelle einer SESAM/SQL-Datenbank laden. Die Eingabedatei ist eine SAM-Datei mit variabler Satzlänge. Folgende Dateien können als Eingabedatei verwendet werden:

- eine Eingabedatei, deren Format SESAM/SQL festlegt (Standardformat)
- eine Eingabedatei, deren Format der Benutzer festlegt (selbst-definiertes Format)
- eine Transferdatei
- eine Eingabedatei im LOAD_FORMAT
- eine Eingabedatei im Delimiter-Format
- eine Eingabedatei im CSV-Format

Der Coded Character Set Name (CCSN) der Eingabedatei wird zur Prüfung gegen den codierten Zeichensatz der Datenbank verwendet.
Im DELIMITER_FORMAT, im CSV_FORMAT und im selbst definierten Format werden zu ladende Zeichenketten nötigenfalls in den codierten Zeichensatz der Datenbank konvertiert.

Fehlerdatei bei LOAD

In der USING FILE-Klausel der LOAD-Anweisung kann der Anwender den Namen für eine Fehlerdatei angeben.

Die Fehlerdatei wird von SESAM/SQL nur bei Auftreten eines Fehlers angelegt bzw. verwendet. Eine bereits vorhandene Fehlerdatei wird von LOAD fortgeschrieben. Die Einträge aus verschiedenen LOAD-Anweisungen sind durch zwei Überschriftszeilen getrennt.

Ohne Angabe der USING FILE-Klausel heißt die Fehlerdatei *catalog.space*.EXC.L

catalog Namen der Datenbank.

space Namen des Anwender-Space, auf dem die Tabelle liegt, in die geladen werden soll.

Bei einer partitionierten Tabelle wird der Spacename der ersten Partition verwendet. Die Fehlerdatei enthält die Fehlerinformationen für alle Partitionen der Tabelle.

EXC.L Defaultwert des Namens der Fehlerdatei für LOAD Anweisungen.

Unterbrechung des Logging bei LOAD OFFLINE

Liegt eine Basistabelle auf einem Anwender-Space, für den Logging vereinbart ist, so protokolliert SESAM/SQL alle DML- bzw. DDL-Anweisungen, die den Tabelleninhalt bzw. die Tabellendefinition ändern, auf DA-LOG- bzw. CAT-LOG-Dateien. Das Logging wird unterbrochen, wenn mit LOAD OFFLINE Daten in diese Tabelle geladen werden. Bei LOAD ONLINE wird das Logging fortgeführt. Unmittelbar nach LOAD OFFLINE mit erfolgreichem CONSTRAINT CHECK muss deshalb der Datenbankverwalter mit der Utility-Anweisung COPY einen SESAM-Sicherungsbestand des Anwender-Space erstellen, auf dem die Tabelle liegt (siehe Seite 368). Andernfalls verbleibt der Anwender-Space im Zustand „copy pending“ und kann nur mit Utility-Anweisungen oder Retrival-Anweisungen bearbeitet werden (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“).

9.3.2 Anwenderdaten entladen mit UNLOAD

Mit der Utility-Anweisung UNLOAD kann der Datenbankverwalter Anwenderdaten aus einer Tabelle einer SESAM/SQL-Datenbank in eine SAM-Datei mit variabler Satzlänge schreiben. Dabei kann der Datenbankverwalter wählen,

- ob SESAM/SQL das Format festlegt (Standardformat)
- ob er das Format für die Ausgabedatei selbst festlegt (selbst-definiertes Format)
- ob die Ausgabedatei im Delimiter-Format erstellt wird
- ob die Ausgabedatei im CSV-Format erstellt wird
- ob die Ausgabedatei als Transferdatei erstellt wird
- ob die Ausgabedatei so erstellt wird, dass sie als Eingabedatei im UNLOAD_FORMAT für die Utility-Anweisung LOAD verwendet werden kann.

Die Ausgabedatei wird mit Coded Character Set Name (CCSN) angelegt. Im DELIMITER_-FORMAT, im CSV-Format und im selbst definierten Format werden die auszugebenden Daten nötigenfalls in den codierten Zeichensatz der Ausgabedatei konvertiert. Im CSV-Format können auch die Spaltennamen (in der ersten Zeile) ausgegeben werden.

Sie können ganze Tabellen (UNLOAD TABLE) oder auch Daten aus ausgewählten Spalten der Tabelle ausgeben (UNLOAD DATA).

Mit UNLOAD ONLINE können Sie Anwenderdaten von Basistabellen oder Views ausgeben. Die auszugebenden Daten können durch eine Suchbedingung eingeschränkt und eine Sortierreihenfolge für die Ausgabedatei kann definiert werden.

Bei UNLOAD OFFLINE können Sie Anwenderdaten von Basistabellen ausgeben. Außerdem können Sie Daten ausgeben:

- aus einem Anwender-Space, für den die Reparatur nicht vollständig durchgeführt werden konnte (Klausel FROM SPACE, Zustand des Anwender-Space: „recover pending“)
- aus einem SESAM-Sicherungsbestand (Klausel FROM COPY_FILE)

Fehlerdatei bei UNLOAD

SESAM/SQL legt beim UNLOAD bei Bedarf eine neue Datei an oder schreibt die bestehende Datei fort. Der Dateiname lautet:

catalog.space.EXC.U bei UNLOAD OFFLINE bzw.
datei.EXC.U bei UNLOAD ONLINE

catalog Name der Datenbank.

space Name des Anwender-Space, auf dem die Tabelle liegt, von der ausgegeben werden soll. Bei einer partitionierten Tabelle wird der Spacename der ersten Partition verwendet. Die Fehlerdatei enthält die Fehlerinformationen für alle Partitionen der Tabelle.

datei Name der Datei, die in der Klausel INTO FILE angegeben ist.

Eine bereits existierende Fehlerdatei wird fortgeschrieben. Die Einträge aus verschiedenen UNLOAD-Anweisungen sind durch eine Überschriftszeile getrennt. Wenn eine Fehlerdatei mindestens einen Satz enthält, wird eine Warnung ausgegeben.

9.3.3 Anwenderdaten zwischen Basistabellen übertragen

Mit den Utility-Anweisungen UNLOAD und LOAD ist es möglich, Anwenderdaten von einer Basistabelle oder einem View in eine andere Basistabelle zu übertragen.

Um beispielsweise Daten von einer Basistabelle A in eine Basistabelle B zu übertragen, entlädt der Anwender für die Basistabelle A die Daten von A mit UNLOAD, Option LOAD_-FORMAT. SESAM/SQL baut dann die Ausgabedatei so auf, dass sie der Anwender für B unmittelbar mit LOAD, Option UNLOAD_FORMAT als Eingabedatei für B verwenden kann.

Ist für eine zu entladende Basistabelle A ein Primärschlüssel definiert, so ist die mit UNLOAD im LOAD_FORMAT erstellte Ausgabedatei nach Primärschlüsselwerten sortiert. Stimmt der Primärschlüssel der Tabelle B, in die zugeladen werden soll, mit dem Primärschlüssel von A überein, dann lässt sich der Zuladevorgang beschleunigen, indem in B mit LOAD SORTED zugeladen wird.

Die beiden Tabellen, zwischen denen die Anwenderdaten übertragen werden sollen, müssen bezüglich der zu entladenden/ladenden Spalten identisch definiert sein.

9.4 Korrektheit der Daten prüfen

SESAM/SQL bietet dem Datenbankverwalter die Möglichkeit, die Korrektheit der Daten unter folgenden Gesichtspunkten zu prüfen:

- Prüfung, ob Integritätsbedingungen erfüllt sind, d.h. Prüfung der Daten auf inhaltliche Korrektheit.
- Prüfung von Tabellen und Indizes auf formale Korrektheit.

9.4.1 Integritätsbedingungen prüfen

Im laufenden Datenbankbetrieb gewährleistet SESAM/SQL durch Plausibilitätsprüfungen, dass nur solche Änderungen an den Daten vorgenommen werden, die mit den Integritätsbedingungen verträglich sind.

Utility-Anweisung CHECK CONSTRAINTS

Mit der Utility-Anweisung CHECK CONSTRAINTS kann der Datenbankverwalter die Einhaltung von Integritätsbedingungen prüfen.

Mit CHECK CONSTRAINTS können geprüft werden:

- ausgewählte Integritätsbedingungen
- alle Integritätsbedingungen, die auf einer oder mehreren Basistabellen definiert sind
- alle Integritätsbedingungen, die auf den Basistabellen eines oder mehrerer Anwender-Spaces definiert sind

Dabei können wahlweise die Integritätsbedingungen für alle angegebenen Tabellen geprüft werden oder nur für die Tabellen, die sich im Zustand „check pending“ befinden (siehe Abschnitt „Space-Zustände nach der Ausführung von Utility-Anweisungen“ auf Seite 166).

Eine Prüfung der Integritätsbedingungen mit CHECK CONSTRAINTS ist immer notwendig, wenn sich nach folgenden Utility-Anweisungen der Datenbestand geändert hat:

- LOAD, wenn LOAD ohne die Klausel CONSTRAINT CHECK ausgeführt wurde
- RECOVER, wenn nicht der aktuelle Zustand der Datenbank wiederhergestellt wurde

In diesen Fällen versetzt SESAM/SQL die von der Utility-Anweisung betroffenen Tabellen bzw. den bearbeiteten Anwender-Space in den Zustand „check pending“.

Wie mit Tabellen zu verfahren ist, die nach Ausführung der Utility-Anweisung CHECK CONSTRAINTS im Zustand „check pending“ verbleiben, ist im Abschnitt „Tabellen im Zustand „check pending“ mit SQL-Anweisungen bearbeiten“ auf Seite 365 beschrieben.

9.4.2 Formale Korrektheit von Tabellen und Indizes prüfen

Der Datenbankverwalter kann die formale Korrektheit von Anwenderdaten und Indizes mit der Utility-Anweisung CHECK FORMAL prüfen. Die Prüfung der formalen Korrektheit umfasst im Wesentlichen die Prüfung SESAM/SQL-interner Blöcke und Tabellen.

Mit CHECK FORMAL prüft der Datenbankverwalter wahlweise eine einzelne Basistabelle, einen einzelnen Index oder alle Basistabellen und Indizes eines Anwender-Space.

CHECK FORMAL wird auch zugelassen auf Replikate und auf SESAM-Sicherungsbestände, die im SQL-Datenbankverzeichnis enthalten sind. Ein CHECK FORMAL auf Fremdkopien ist möglich, wenn die Fremdkopie als Datenbank im SQL-Datenbankverzeichnis eingetragen wird. Damit sie ihre Eigenschaft als Fremdkopie nicht verliert, muss sie mit ACCESS=READ eingetragen werden.

Die Datenbank, deren Spaces, Tabellen oder Indizes geprüft werden sollen, muss im SQL-Datenbankverzeichnis des DBH mit der Eigenschaft ADMIN eingetragen sein, damit defekte Spaces auch als defekt markiert werden.

Ist die Datenbank mit ACCESS=READ, als Sicherungsbestand oder als Replikat eingetragen, wird ein fehlerhafter Space nicht als defekt markiert. Hinweise auf den Fehler erhalten Sie dann nur im SQLSTATE und in der Fehlerdatei.

Fehlerdatei für CHECK FORMAL

SESAM/SQL legt für CHECK FORMAL eine Fehlerdatei an oder schreibt die bestehende fort, wenn CHECK FORMAL Inkonsistenzen feststellt.

Der Dateiname in der BS2000-Benutzerkennung des DBH lautet: *catalog*.*space*.EXC.C

catalog Name der Datenbank

space Name des Anwender-Space, auf dem die Formalprüfung durchgeführt wird

Bei CHECK FORMAL TABLE für eine partitionierte Tabelle wird für jede fehlerhafte Partition eine Fehlerdatei mit dem Spacenamen der Partition gebildet.

EXC.C Defaultwert des Namens der Fehlerdatei für CHECK FORMAL-Anweisungen

Zu jeder entdeckten Inkonsistenz trägt SESAM/SQL einen Satz in die Fehlerdatei ein. Die Fehlerdatei wird von SESAM/SQL nur dann angelegt bzw. verwendet, wenn tatsächlich Inkonsistenzen festgestellt werden.

9.5 Gesperrte Anwender-Spaces bearbeiten

Eine Utility-Anweisung kann einen Space in einem Zustand zurücklassen, in dem er nicht uneingeschränkt weiter bearbeitet werden kann (siehe Kapitel „Utility-Konzept“ auf Seite 159).

Welche Utility-Anweisungen in einem bestimmten Zustand eines Anwender-Space ausgeführt werden dürfen und in welchen neuen Zustand der Anwender-Space dadurch jeweils übergeführt wird, ist im Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“ ausführlich beschrieben.

- Spaces im Zustand „copy pending“ erfordern COPY.
- Spaces im Zustand „load running“ erfordern RECOVER und ggf. anschließend RECOVER INDEX.
- Spaces im Zustand „recover pending“ erfordern RECOVER.
- Spaces im Zustand „check pending“ erfordern CHECK CONSTRAINTS. Wenn CHECK CONSTRAINTS Integritätsverletzungen meldet, müssen diese Verletzungen beseitigt werden.

Tabellen im Zustand „check pending“ mit SQL-Anweisungen bearbeiten

Tabellen eines Anwender-Space, der sich im Zustand „check pending“ befindet (siehe Seite 166), können von den Eigentümern dieser Tabellen mit SQL-Anweisungen zur Datenmanipulation bearbeitet werden. Auf diese Weise lassen sich alle Sätze anpassen oder löschen, die eine Integritätsbedingung verletzen. Voraussetzung ist, dass diese SQL-Anweisungen mit dem Pragma CHECK OFF eingegeben werden. Das Pragma CHECK OFF ist, syntaktisch gesehen, ein gewöhnlicher SQL-Kommentar. Es ermöglicht jedoch die Ausführung von SQL-Anweisungen auf Tabellen, für die Integritätsbedingungen verletzt sind. Innerhalb einer SQL-Anweisung wird ein Pragma durch die Zeichenfolge „--%“ eingeleitet und durch Zeilenende abgeschlossen.
Ausführlich beschrieben ist das Pragma CHECK OFF im Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“.

Verletzungen von Integritätsbedingungen lassen sich auch beheben, indem die Eigentümer der betroffenen Tabellen die verletzten Integritätsbedingungen mit der SQL-Anweisung ALTER TABLE ... DROP CONSTRAINT löschen.

Nachdem alle Verletzungen von Integritätsbedingungen behoben sind, muss der Datenbankverwalter erneut die Utility-Anweisung CHECK CONSTRAINTS ausführen, um die betroffenen Anwender-Spaces bzw. Tabellen für die berechtigten Personengruppen wieder allgemein verfügbar zu machen.

9.6 Datenbank warten

Im Zusammenhang mit der Wartung einer SESAM/SQL-Datenbank fallen folgende Tätigkeiten an:

- Spaces und Basistabellen auf Anwender-Spaces reorganisieren
- Aufgaben im Rahmen des Media-Recovery erledigen
- Datenbanken und Tabellen umstellen
- Metadaten abfragen

9.6.1 Spaces und Basistabellen reorganisieren

Mit der Utility-Anweisung REORG kann der Datenbankverwalter den Catalog-Space, einzelne Anwender-Spaces oder auch einzelne Basistabellen auf Anwender-Spaces reorganisieren. Bei der Reorganisation speichert SESAM/SQL logisch zusammengehörige Speicherbereiche auch physikalisch zusammenhängend ab.

Zweck der Reorganisation

Die Blöcke des Catalog-Space, der einzelnen Anwender-Spaces und der Basistabellen sind jeweils in logischer Reihenfolge verkettet. Zu Beginn der Bearbeitung stimmen physikalische Reihenfolge (nach aufsteigenden Blocknummern) und logische Reihenfolge (inhaltlich, entlang der Verkettung) der Blöcke überein.
Im Verlauf des Datenbankbetriebs kann SESAM/SQL neue Blöcke anlegen, wenn der freie Platz in den bereits vorhandenen Blöcken für neu hinzukommende Daten nicht mehr ausreicht. Die logische Reihenfolge neuer und alter Blöcke ist dann durch die Verkettung gesichert, stimmt aber nicht mehr mit der physikalischen Reihenfolge überein. Dies kann bei sequenzieller Abarbeitung eines Space die Zugriffsgeschwindigkeit erheblich beeinträchtigen.
Die Reorganisation stellt die Übereinstimmung von logischer und physikalischer Reihenfolge der Blöcke wieder her.

Bei der Reorganisation berücksichtigt SESAM/SQL auch die Freiplatzreservierung, die der Datenbankverwalter zuvor für den betreffenden Space (Catalog-Space oder Anwender-Space) mit den Anweisungen CREATE CATALOG, CREATE SPACE bzw. ALTER SPACE in der Klausel PCTFREE vereinbart hat.

Utility-Anweisung REORG [CATALOG_]SPACE

Wenn der Datenbankverwalter zuvor mit der SQL-Anweisung ALTER SPACE für den zu reorganisierenden Anwender-Space eine neue Storage Group angegeben hat, dann erfolgt bei der Reorganisation die physikalische Verlagerung des Anwender-Space auf diese Storage Group, wenn nicht durch die Angabe von COPY oder die Angabe einer Arbeitsdatei eine andere Festlegung erfolgt.

SESAM/SQL reorganisiert den Space in eine Arbeitsdatei. Die Arbeitsdatei kann vorgegeben werden. Sonst verwendet SESAM/SQL eine Standard-Arbeitsdatei, deren Name aus dem Namen der Space-Datei durch Ergänzung mit dem Suffix .REORG gebildet wird. Die Standard-Arbeitsdatei kann auch vom Anwender angelegt werden; sie wird dann mit den definierten Dateimerkmalen als Standard-Arbeitsdatei verwendet.

Mit der Anweisungsfolge EXPORT TABLE, DROP TABLE und IMPORT TABLE können Sie einen Space, der Tabellen und Indizes enthält, so umstrukturieren, dass Indizes und Tabellen auf separaten Spaces liegen. Zunächst wird die betroffene Tabelle mit den dazugehörigen Indizes in eine Exportdatei exportiert. Die Tabelle selbst wird mit DROP TABLE gelöscht. Anschließend wird die Tabelle mit IMPORT TABLE aus der Export-Datei wieder auf den ursprünglichen Space importiert. Für die Indizes kann dabei ein separater Index-Space angegeben werden (siehe Abschnitt „Basistabellen importieren mit IMPORT TABLE“ auf Seite 357).

Eine einfachere Vorgehensweise ist möglich, wenn auf einem reinen Tabellen-Space von SESAM/SQL Indizes zu Eindeutigkeitsbedingungen angelegt worden sind. Diese Indizes können dadurch auf andere Spaces verlagert werden, dass die Indizes explizit mit CREATE INDEX auf anderen Spaces angelegt werden. SESAM/SQL verlegt dann die Eindeutigkeitsbedingung (UNIQUE) auf den explizit definierten Index und löscht den implizit definierten.

Utility-Anweisung REORG ONLINE TABLE

SESAM/SQL führt die Reorganisation einer Basistabelle durch Modifizieren und Kopieren der Blöcke innerhalb des Anwender-Space durch. Da hierfür keine exklusiven Transaktionssperren benötigt werden, können andere DML-Anwendungen lesend und ändernd auf die Basistabelle zugreifen.

Bei partitionierten Tabellen kann auch eine einzelne Partition reorganisiert werden.
Die Partition wird durch die Klausel ON SPACE festgelegt. Sonst werden nacheinander alle Partitionen der Basistabelle reorganisiert.

9.6.2 Aufgaben im Rahmen des Media-Recovery

Im Abschnitt „Media Recovery“ (siehe Seite 211) wird das Konzept des Media-Recovery erläutert. Der vorliegende Abschnitt geht näher auf die Tätigkeiten ein, die im Rahmen des Media-Recovery anfallen und stellt die Utility-Anweisungen vor, mit denen der Datenbankverwalter die einzelnen Tätigkeiten durchführt.

9.6.2.1 SESAM-Sicherungsbestand erstellen mit COPY

Mit der Utility-Anweisung COPY kann der Datenbankverwalter SESAM-Sicherungsbestände der gesamten SESAM/SQL-Datenbank oder von Teilen der Datenbank wie Catalog-Space und Anwender-Spaces erstellen. Die SESAM-Sicherungsbestände legt SESAM/SQL wahlweise auf Magnetbandkassette oder Platte an. Das Erstellen eines SESAM-Sicherungsbestandes kann im Modus online oder im Modus offline erfolgen. Auch die Sicherung mit HSMS von Spiegelplatten eines lokalen oder entfernten Storage-Systems ist möglich. Ferner kann der Datenbankverwalter den SESAM-Sicherungsbestand auf formale Korrektheit prüfen und veranlassen, dass ein Anwender-Space oder der Catalog-Space nach der Erstellung der Kopie mit Logging betrieben werden. Spaces, auf denen nur Indizes liegen und die nicht in der logischen Datensicherung sind, können von der Sicherung ausgenommen werden.

Liegt die Datenbank in einer DB-Kennung, so versucht SESAM/SQL immer zuerst, den SESAM-Sicherungsbestand in der DB-Kennung anzulegen. Das ist nur möglich, wenn die Vorbereitungen für das Anlegen der Dateien getroffen worden sind, siehe Abschnitt „Datenbankdateien und Jobvariablen auf fremden Benutzerkennungen“ auf Seite 308. Andernfalls werden die Kopien in der DBH-Kennung angelegt.

Falls das Logging eingeschaltet ist, ist mit COPY CATALOG und COPY CATALOG_SPACE stets ein Wechsel der CAT-LOG-Datei und DA-LOG-Datei verbunden.
Bei Datenbanken, die ohne Logging betrieben werden, richtet SESAM/SQL beim ersten COPY CATALOG bzw. COPY CATALOG_SPACE die CAT-REC-Datei gemäß den Angaben in der Medientabelle ein.
Als Aufsetzpunkt für Replikate wird bei jedem COPY CATALOG und COPY CATALOG_SPACE eine CAT-REC-Kopie auf den gleichen Medien angelegt wie die CAT-LOG-Datei. Es werden keine Metadaten über diese zusätzlichen Kopien hinterlegt. Soll die CAT-REC-Kopie auf der DB-Kennung liegen, so gilt das oben Ausgeführte. Eine von einem früheren COPY vorhandene CAT-REC-Kopie wird überschrieben.

SESAM-Sicherungsbestände auf Magnetbandkassette können über die Utility-Anweisung COPY ... USING DIRECTORY wahlweise mit Hilfe der Softwareprodukte HSMS oder ARCHIVE erstellt werden (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“). Ist eine HSMS-Parameterdatei vorhanden, wird ein für COPY ... USING DIRECTORY angegebener Name als HSMS-Archiv betrachtet. Andernfalls erfolgt die Sicherung mit ARCHIVE.

SESAM-Sicherungsbestand online oder offline erstellen

Bei der Ausführung von COPY erzeugt SESAM/SQL zunächst einen transaktionslosen Zustand auf den von COPY betroffenen Spaces (Catalog-Space bzw. Anwender-Spaces).

- Bei COPY ONLINE sind folgende Effekte zu beachten:
  SESAM/SQL lässt mit Beginn der Kopieerstellung wieder SQL-Anweisungen zum Abfragen und Ändern von Daten sowie CALL-DML-Anweisungen zu. Alle anderen SQL-Anweisungen und alle Utility-Anweisungen weist SESAM/SQL weiterhin mit SQL-Statuscode ab.
  Soll die Online-Sicherung in ein HSMS-Archiv erfolgen, so dürfen die zu sichernden Datenbank-Dateien nicht auf Privatplatte liegen.
  Alle Blöcke der zu kopierenden Anwender-Spaces, die während des Kopiervorgangs geändert werden, protokolliert SESAM/SQL vor ihrer Änderung als sog. physikalische Before Images (PBI) in der PBI-Datei (bei Plattenkopie und ARCHIVE) bzw. sie werden von HSMS in die HSMS-Arbeitsdatei protokolliert. Diese Datei wird immer in der Benutzerkennung angelegt, in der der DBH abläuft. Mit Abschluss des Kopiervorgangs beendet SESAM/SQL diese PBI-Protokollierung und gibt die betroffenen Anwender-Spaces wieder für alle Anweisungen frei.
  Bei einem SESAM-Sicherungsbestand auf Platte spielt SESAM/SQL die protokollierten Before Images sofort in den SESAM-Sicherungsbestand ein, so dass dieser SESAM-Sicherungsbestand denselben Stand aufweist wie der zugehörige Anwender-Space im transaktionsfreien Zustand unmittelbar vor dem eigentlichen Kopiervorgang.
  Bei einem SESAM-Sicherungsbestand auf Magnetbandkassette mit ARCHIVE sichert SESAM/SQL die PBI-Datei ebenfalls auf eine separate Magnetbandkassette. Das Einspielen der Datei erfolgt in diesem Fall erst bei der Reparatur bzw. bei dem Zurücksetzen mit der Utility-Anweisung RECOVER.
  Bei der Sicherung in ein HSMS-Archiv wird der zu sichernde Block von der Datenbankdatei oder von der HSMS-Arbeitsdatei gelesen und anschließend gesichert.
  Von einem bestimmten SESAM/SQL-DBH können gleichzeitig mehrere Utility-Anweisungen COPY ONLINE ausgeführt werden. Sie müssen sich auf die Spaces unterschiedlicher Datenbanken beziehen.
- Bei COPY OFFLINE hält SESAM/SQL sämtliche ändernden Zugriffe auf die von COPY OFFLINE betroffenen Anwender-Spaces bzw. den Catalog-Space zurück bis der COPY OFFLINE abgeschlossen ist.
  Der Stand des SESAM-Sicherungsbestandes entspricht dem Zustand des zugehörigen Space bzw. der zugehörigen Spaces im transaktionskonsistenten Zustand unmittelbar vor dem eigentlichen Kopiervorgang.

SESAM-Sicherungsbestand mit HSMS von Spiegelplatten erstellen

Mit `COPY ... USING DIRECTORY` *hsms_archiv_name* { `BY_ADD_MIRROR_UNIT` | `BY_SRDF_TARGET` }
können Datenbank-Dateien, die auf einer Spiegelplatte liegen, sehr performant in ein HSMS-Archiv auf Platte oder Magnetbandkassette gesichert werden. Abhängig von der örtlichen Lage der Spiegelplatte sind zwei Fälle zu unterscheiden (hier am Beispiel der Symmetrix-Funktion TimeFinder/Mirror erläutert):

1. BY_ADD_MIRROR_UNIT
   In diesem Fall wird im laufenden Betrieb die Funktion TimeFinder/Mirror (Symmetrix Multi Mirror Facility) der Symmetrix-Systeme genutzt. Der Datenbankbetrieb läuft auf der sogenannten Normal-Unit in der Local-Symmetrix. Zur Datenspiegelung wird eine zusätzliche Platte in der Local-Symmetrix, die sogenannte Additional-Mirror-Unit, verwendet. Das Paar Normal-Unit und Additional-Mirror-Unit wird als Multi-Mirror-Paar bezeichnet.

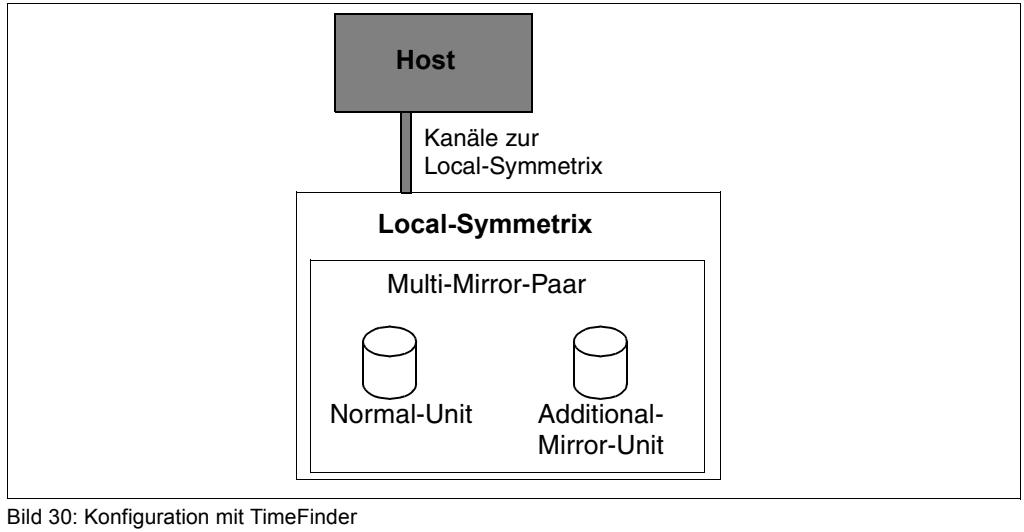


Bild 30: Konfiguration mit TimeFinder

2. BY_SRDF_TARGET
   In diesem Fall wird im laufenden Betrieb die Funktion SRDF (Symmetrix Remote Data Facility) der Symmetrix-Systeme genutzt. Der Datenbankbetrieb läuft auf der sogenannten Source-Unit in der Local-Symmetrix. Zur Datenspiegelung wird eine zusätzliche Platte in der (entfernten) Remote-Symmetrix, die sogenannte Target-Unit, verwendet. Das Paar Source-Unit und Target-Unit wird als Remote-Copy-Paar bezeichnet. Das Remote-Copy-Paar muss im synchronen Verarbeitungsmodus betrieben werden. Das HSMS-Archiv muss auf dem gleichen System liegen, an das die Source-Unit angeschlossen ist.

Zum Erstellen eines SESAM-Sicherungsbestandes ist es erforderlich, dass der Target-Unit in der Remote-Symmetrix eine Additional-Mirror-Unit zugewiesen ist.
D.h. Target-Unit und Additional-Mirror-Unit liegen beide in der Remote-Symmetrix und bilden dort ein Multi-Mirror-Paar (TimeFinder/Mirror).

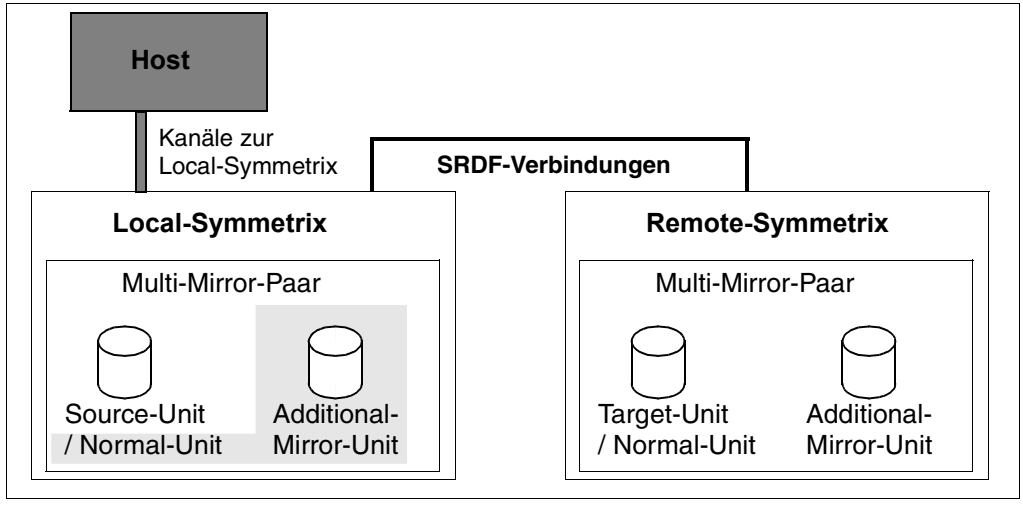


Bild 31: Konfiguration mit SRDF und TimeFinder

Bemerkung: Zur weiteren Datenspiegelung in der Local-Symmetrix kann zur Source-Unit zusätzlich eine Additional-Mirror-Unit (TimeFinder/Mirror) betrieben werden. Auch von ihr könnte mit BY_ADD_MIRROR_UNIT ein SESAM-Sicherungsbestand erstellt werden (im Bild grau hinterlegt, in der weiteren Betrachtung ohne Bedeutung).

In beiden Fällen wird der SESAM-Sicherungsbestand von der Additional-Mirror-Unit mit HSMS erstellt. Die Datenbankdateien müssen auf derselben Additional-Mirror-Unit liegen.

Soweit in der folgenden Beschreibung von Spiegelplatten gesprochen wird, sind Additional-Mirror-Units (Symmetrix-Systeme) oder Clone-Units (ETERNUS DX-, Symmetrix- oder CLARiiON CX-Systeme) gleichermaßen gemeint.

Nähere Informationen zu diesen Storage-Systemen sowie zu ihren Replikations-Funktionen finden Sie im Handbuch „SHC-OSD (BS2000)“.

Ablauf der Sicherung mit HSMS von Spiegelplatten

Die Sicherung erfolgt in drei Phasen:

- Zuerst wird die Spiegelplatte von der Normal-Unit (Fall 1 im vorangehenden Abschnitt) bzw. der Target-Unit (Fall 2) getrennt. Dabei sind lesende DML-Zugriffe auf die Datenbank möglich.
- Die angegebenen Datenbankdateien werden von der Spiegelplatte in das angegebene HSMS-Archiv gesichert.
  Die Datenbank auf der Normal-Unit (Fall 1) bzw. der Source-Unit (Fall 2) wird vom DBH prozessiert. Die Spiegelung auf die Target-Unit (Fall 2) wird fortgeführt. Während der Dauer der Sicherung sind lesende Zugriffe (bei COPY … OFFLINE) oder auch ändernde DML-Zugriffe (bei COPY … ONLINE) auf die Datenbank möglich.
- Nachdem die Sicherung in das HSMS-Archiv erfolgt ist, werden die Spiegelplatte und die Normal-Unit (Fall 1) bzw. die Target-Unit (Fall 2) wieder synchronisiert. Auch während dieser Zeit sind lesende oder ändernde Zugriffe möglich.

SESAM/SQL nutzt bei der Ausführung der COPY-Anweisung die Funktion Concurrent Copy (CCOPY) von HSMS. HSMS nutzt seinerseits das Softwareprodukt SHC-OSD. Für die Sicherung in ein HSMS-Archiv baut SESAM/SQL die HSMS Anweisung BACKUP-FILES mit dem Parameter WRITE-CHECKPOINTS=*YES auf, um Wiederaufsetzpunkte zu schreiben. Damit kann nach einem Abbruch der HSMS-Sicherungslauf mit der HSMS-Anweisung RESTART-REQUESTS fortgesetzt werden (weitere Informationen dazu finden Sie im Handbuch „HSMS (BS2000)“, Band 1).

Der Vorteil dieses Verfahrens ist, dass der Datenbankverwalter sich weder um die Trennung noch um die Synchronisation der Spiegelplatte kümmern muss.

Für diese Funktion benötigt die DBH-Task eines der Privilegien TSOS oder HSMS-ADMINISTRATION.

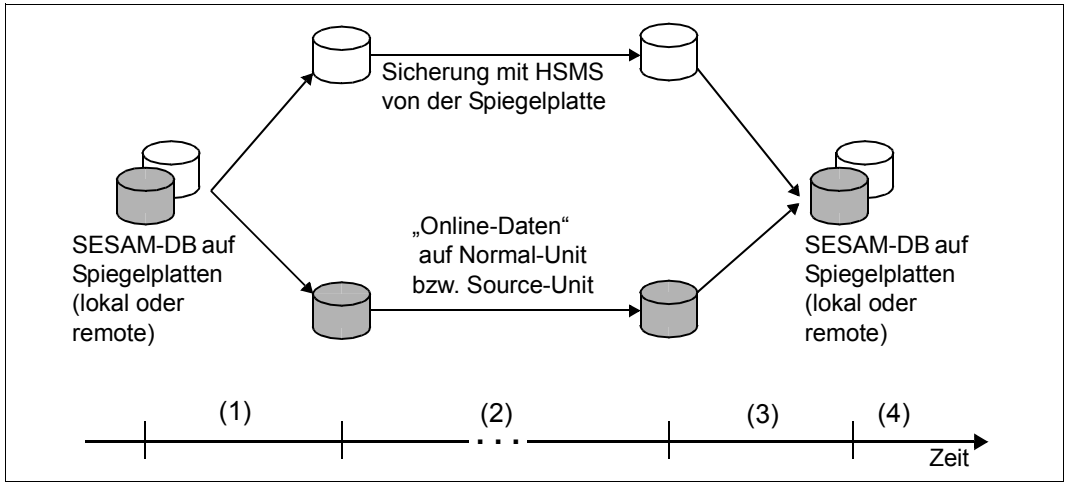


Bild 32: Ablauf der Sicherung mit HSMS von Additional-Mirror-Units

(1) Initialisierungsphase: Zeit vom Beginn der COPY-Anweisung bis zur Trennung der Spiegelplatten. In dieser Phase sind nur lesende Zugriffe auf die zu sichernden Spaces möglich. Ändernde Zugriffe warten auf das Ende der Initialisierungsphase. SESAM/SQL baut die HSMS-Anweisung BACKUP-FILES auf.

(2) Sicherungs- und Betriebsphase: Zeit, in der HSMS-Sicherung und Datenbankbetrieb parallel laufen. HSMS sichert die Datenbankdateien von der Spiegelplatte in ein HSMS-Archiv. Die Datenbank auf der Normal-Unit bzw. der Source-Unit wird parallel dazu vom DBH prozessiert. Ändernde DML-Anweisungen sind bei COPY ONLINE wieder zugelassen.

(3) Synchronisationsphase: Zeit vom Ende der HSMS-Sicherung bis zum Abschluss der Synchronisation der Spiegelplatten. Nach der HSMS-Sicherung werden die Spiegelplatten wieder synchronisiert. Lesende und ändernde DML-Anweisungen sind weiter möglich.

(4) Die Sicherung war erfolgreich, wenn die Sicherung in das HSMS-Archiv erfolgreich war und die Formalprüfung bei Angabe des Parameters CHECK FORMAL zu keinem Fehler geführt hat. Die erfolgreiche Sicherung wird in die RECOVERY_UNITS eingetragen.

Hinweise zur Sicherung mit dem Parameter CHECK FORMAL

Bei einer Online-Sicherung mit HSMS von Spiegelplatten kann der Parameter CHECK FORMAL angegeben werden.
Wenn die Source- oder Normal-Unit zu einem shared Pubset gehört, dann muss der SESAM/SQL-DBH auf dem Server ablaufen, der für das shared Pubset die Master-Seite darstellt.
Die formale Prüfung auf den zu sichernden Spaces wird auf der abgetrennten Additional-Mirror-Unit parallel zur Sicherung der Dateien in das HSMS-Archiv durchgeführt.

Sonst darf der Parameter bei einer Online-Sicherung auf Magnetbandkassette nicht angegeben werden.

Wird bei einer Offline-Sicherung auf Magnetbandkassette der Parameter CHECK FORMAL angegeben, so wird zuerst die formale Prüfung auf den zu sichernden Spaces durchgeführt und anschließend der Auftrag an HSMS gegeben. Dies führt dazu, dass die zu sichernden Spaces so lange gesperrt sind bis die formale Prüfung abgeschlossen ist. Erst dann wird mit der Trennung der Platten begonnen.

Deshalb sollten Sie bei entsprechenden Voraussetzungen der Online-Sicherung mit HSMS von Spiegelplatten den Vorzug geben.
Sonst wird bei einer Offline-Sicherung folgendes Vorgehen empfohlen:
Die Sicherung sollte mit NO CHECK FORMAL erzeugt werden. Anschließend sollte der Datenbankverwalter die Sicherung aus dem HSMS-Archiv wieder einspielen, die Datenbank in das SQL-Datenbankverzeichnis eines anderen DBH eintragen und die formale Prüfung durchführen.

Wird bei einer Formalprüfung ein Fehler gefunden, so war die Sicherung nicht erfolgreich; der Originalspace wird aber nicht als defekt gekennzeichnet.

9.6.2.2 Metadaten von SESAM-Sicherungsbeständen pflegen

Die Catalog-Tabellen RECOVERY_UNITS und DA_LOGS liegen auf dem Catalog-Space der Datenbank. RECOVERY_UNITS enthält Einträge zu den SESAM-Sicherungsbeständen der Anwender-Spaces, in DA_LOGS sind Informationen zu den DA-LOG-Dateien abgelegt.
Im Abschnitt „Dateien und Tabellen für das Media-Recovery“ auf Seite 212 sind die Catalog-Tabellen RECOVERY_UNITS und DA_LOGS ausführlich beschrieben.

Die CAT-REC-Datei (Catalog-Recovery-Datei) enthält Recovery-Unit-Sätze für den Catalog-Space und dazu gehörige CAT-LOG-Sätze.
Im Abschnitt „CAT-REC-Datei“ auf Seite 215 ist die CAT-REC-Datei ausführlich beschrieben.

Mit der Utility-Anweisung MODIFY kann der Datenbankverwalter

- in der Catalog-Tabelle RECOVERY_UNITS Sätze löschen, die sich auf SESAM-Sicherungsbestände eines bestimmten Anwender-Space beziehen.

  Mindestens die letzten beiden Sätze bleiben immer erhalten. Einer der verbleibenden Sätze bezieht sich auf den zuletzt erstellten SESAM-Sicherungsbestand des Anwender-Space. Ansonsten werden für diesen Anwender-Space wahlweise alle übrigen Sätze oder Sätze eines bestimmten Alters gelöscht.

- in den Catalog-Tabellen RECOVERY_UNITS und DA_LOGS Sätze unabhängig von ihrer Zuordnung zu bestimmten Anwender-Spaces löschen.

  Erhalten bleiben in RECOVERY_UNITS pro Anwender-Space mindestens die letzten beiden Sätze. Einer der verbleibenden Sätze muss sich auf den zuletzt erstellten SESAM-Sicherungsbestand des Anwender-Space beziehen. Ansonsten werden wahlweise alle übrigen Sätze oder Sätze eines bestimmten Alters aus RECOVERY_UNITS gelöscht.
  In DA_LOGS bleiben jeweils nur die Sätze zu DA-LOG-Dateien erhalten, die an SESAM-Sicherungsbestände anschließen, für die nach Ausführung von MODIFY noch Sätze in RECOVERY_UNITS verblieben sind.

  Fremdkopien stellen in diesem Zusammenhang einen Sonderfall dar. Bei der Arbeit mit ihnen entstehen keine neuen Einträge in der Tabelle RECOVERY_UNITS. Um aus der Catalog-Tabelle DA_LOGS Sätze löschen zu können, kann die Klausel UNRESTRICTED angegeben werden. Es werden dann Sätze aus den DA_LOGS gelöscht, ohne Rücksicht auf einen letzten RECOVERY_UNITS Satz.

ACHTUNG!
Mit UNRESTRICTED werden in den RECOVERY_UNITS und DA_LOGS alle Sätze mit dem angegebenen Alter gelöscht, bis auf den jeweils letzten. Dieser bleibt aus technischen Gründen erhalten.
Es ist kein RECOVER mehr möglich, wenn die dafür benötigten Sätze in den Catalog-Tabellen RECOVERY_UNITS und DA_LOGS gelöscht wurden. Der älteste für eine Fremdkopie eines Space benötigte Satz in DA_LOGS ist der Satz, der beim Schließen des Space nach dem letzten Update auf ihm entstanden ist. Entsteht die Fremdkopie während oder nach einer Session, in der der Space nicht geöffnet wurde oder in der nur gelesen wurde, wird der Aufsetzpunkt im Space nicht aktualisiert.

- in der aktuellen CAT-REC-Datei Recovery-Unit- und CAT-LOG-Sätze löschen, die sich auf SESAM-Sicherungsbestände des Catalog-Space beziehen.

  Erhalten bleibt in der CAT-REC-Datei der jüngste Recovery-Unit-Satz und die dazu gehörenden CAT-LOG-Sätze, damit noch eine Recovery mit SESAM/SQL möglich ist. Ansonsten werden alle übrigen Sätze oder Sätze eines bestimmten Alters gelöscht.

  Fremdkopien stellen in diesem Zusammenhang einen Sonderfall dar. Bei der Arbeit mit ihnen entstehen keine Recovery-Unit-Sätze in der CAT-REC-Datei. CAT-LOG-Sätze können nicht ohne entsprechenden Recovery-Unit-Satz gelöscht werden.

9.6.2.3 CAT-REC-Datei pflegen

Die CAT-REC-Datei enthält Einträge zu den SESAM-Sicherungsbeständen des Catalog-Space sowie Informationen zu den CAT-LOG-Dateien. Im Abschnitt „Dateien und Tabellen für das Media-Recovery“ auf Seite 212 ist die CAT-REC-Datei ausführlich beschrieben.

Mit Hilfe des Utility-Monitors kann sich der Datenbankverwalter über den Inhalt der CAT-REC-Datei informieren. Er kann auch CAT-REC-Dateien online oder offline modifizieren und so Sätze zu nicht mehr benötigten SESAM-Sicherungsbeständen daraus löschen (siehe Handbuch „Utility-Monitor“).

Mit der Utility-Anweisung MODIFY kann der Datenbankverwalter die aktuelle CAT-REC-Datei online modifizieren (siehe Handbuch „SQL-Sprachbeschreibung Teil 2: Utilities“).

Die CAT-REC-Datei sollte mit Dual Recording by Volume oder mit Dual Copy doppelt geführt werden. Ein SESAM-Sicherungsbestand der CAT-REC-Datei kann zwar erstellt werden, wenn die Datenbank nicht im SQL-Datenbankverzeichnis eingetragen ist. Änderungen nach dem Zeitpunkt des Erstellens dieses SESAM-Sicherungsbestandes würden jedoch nicht automatisch protokolliert, so dass die CAT-REC-Datei gegebenenfalls nur auf den Stand dieses SESAM-Sicherungsbestandes zurückgeführt werden könnte.

9.6.2.4 SESAM-Sicherungsbestände und Logging-Dateien im BS2000 löschen

Wenn der Datenbankverwalter mit MODIFY Sätze aus den Catalog-Tabellen RECOVERY_UNITS und DA_LOGS, bleiben die zugehörigen SESAM-Sicherungsbestände der Anwender-Spaces und die DA-LOG-Dateien erhalten. Entsprechendes gilt für nicht mehr benötigte SESAM-Sicherungsbestände des Catalog-Space sowie für nicht mehr benötigte CAT-LOG-Dateien, für die der Datenbankverwalter mit der Utility-Anweisung MODIFY oder mit Hilfe des Utility-Monitors Einträge aus der CAT-REC-Datei gelöscht hat.

Es empfiehlt sich, nicht mehr benötigte SESAM-Sicherungsbestände und Logging-Dateien im BS2000 zu löschen. Auf Platte angelegte Dateien werden mit dem Kommando DELETE-FILE ...,OPTION=*DESTROY-ALL gelöscht. SESAM-Sicherungsbestände auf Magnetbandkassette sind Bestandteil des zugeordneten HSMS-Archivs oder ARCHIVE-Directory und müssen deshalb mit Hilfe der Softwareprodukte HSMS oder ARCHIVE gelöscht werden (siehe Handbücher „HSMS (BS2000)“ und „ARCHIVE (BS2000)“).

9.6.2.5 Reparatur und Zurücksetzen

Reparatur und Zurücksetzen dienen zur Wiederherstellung einer SESAM/SQL-Datenbank nach Fehlersituationen, die sich durch andere Maßnahmen nicht beheben lassen (siehe Abschnitt „Mögliche Fehlersituationen und geeignete Recovery-Maßnahmen“ auf Seite 209). Für Reparatur und Zurücksetzen verwendet der Datenbankverwalter die Utility-Anweisung RECOVER.

Zum Reparieren und Zurücksetzen von Anwender-Spaces, des Catalog-Space und der gesamten Datenbank können auch Fremdkopien statt der mit COPY erzeugten SESAM-Sicherungsbestände eingesetzt werden (siehe Abschnitt „Reparieren und Zurücksetzen mit Fremdkopien“ auf Seite 392).

Ein Reparieren und Zurücksetzen von Anwender-Spaces, des Catalog-Space und der gesamten Datenbank ist auch mit Hilfe eines Replikats möglich. Informieren Sie sich dazu im Abschnitt „Replikat einer Datenbank“ auf Seite 247 oder auch im Abschnitt „Replikat einer Datenbank“ auf Seite 380.

Reparatur und Zurücksetzen aus SESAM-Sicherungsbeständen sind im Rahmen der „Media Recovery“ ab Seite 231 ausführlich beschrieben.

9.6.2.6 Indizes neu aufbauen

Mit der Utility-Anweisung RECOVER INDEX kann der Datenbankverwalter Indizes neu aufbauen bzw. wiederherstellen, siehe Abschnitt „Indizes wiederherstellen“ auf Seite 246.

9.6.3 Datenbanken und Tabellen umstellen

Mit der Utility-Anweisung MIGRATE kann der Datenbankverwalter

- eine SESAM/SQL-Datenbank der Version 1.1 oder einer früheren Version in eine Basistabelle der aktuellen Version umstellen
- eine CALL-DML/SQL-Tabelle in eine SQL-Tabelle umstellen
- eine nur-CALL-DML-Tabelle in eine CALL-DML/SQL-Tabelle umstellen

9.6.4 Metadaten abfragen

Über die Informationsschemata (siehe Seite 84) ermöglicht SESAM/SQL dem Datenbankverwalter Zugang zu den Metadaten der Datenbank.

Die Abfrage der Informationen kann innerhalb eines ESQL-Programms mit SQL-Anweisungen (siehe Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“) oder mit Hilfe des Utility-Monitors (siehe Handbuch „Utility-Monitor“) erfolgen.

Die Tabellen der Informationsschemata sind im Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“ beschrieben.

9.7 Partitionierung einer Basistabelle ändern

Mit der Utility-Anweisung ALTER PARTITIONING FOR TABLE kann der Anwender

- einer Basistabelle mit Primärschlüssel eine Partition hinzufügen (ADD)
- die Partitionsgrenzen einer Partition ändern (ALTER)
- eine Partition löschen (DROP)

Auch einer nicht-partitionierten Basistabelle mit Primärschlüssel kann eine Partition hinzugefügt werden - sie wird dadurch zu einer partitionierten Basistabelle.
Wenn die vorletzte Partition einer partitionierten Tabelle gelöscht wird, dann wird die Tabelle automatisch (wieder) zu einer nicht-partitionierten Tabelle.

Zweck der Partitionierung

Ein stark schwankender Datenbestand in einer Basistabelle erfordert einen flexiblen Umgang mit den gespeicherten Datensätzen. Ein Weg für die Verteilung von Datenbeständen auf unterschiedliche Anwender-Spaces ist die Partitionierung einer Basistabelle, siehe den Abschnitt „Partitionierte Tabelle“ auf Seite 87. Die Partitionierung einer Basistabelle wird bei ihrer Definition mit der CREATE TABLE-Anweisung festgelegt. Häufig werden dabei die Partitionen nach zeitlichen Kriterien, z.B. nach den Monaten des Jahres, gebildet.

Mit der Utility-Anweisung ALTER PARTITIONING FOR TABLE kann der Anwender komfortabel auf geänderte Anforderungen an die Verteilung des Datenbestandes einer Basistabelle reagieren.

9.8 Replikat einer Datenbank

Im Zusammenhang mit dem Replikat einer Datenbank gibt es folgende Anwendungen:

- Replikat erstellen
- Replikat oder Teile des Replikats aktualisieren
- Replikat um Anwender-Spaces erweitern
- Anwender-Spaces, Catalog-Space oder die gesamte Datenbank mit Hilfe eines Replikats reparieren

Die folgende Tabelle gibt einen Überblick über die erforderlichen Tätigkeiten und ordnet den Tätigkeiten die entsprechenden Utility-Anweisungen zu. Die rechte Spalte zeigt die entsprechende Hauptfunktion des Utility-Monitors.

Tätigkeit	**Anweisung und Funktion**	**Hauptfunktion im Utility-Monitor**
Replikat erstellen	– Erzeugen einer Kopie der Datenbank – CREATE REPLICATION	Hauptfunktion COPY & RECOVER / REPLICATION (COP) mit Folgemasken Maske COP.6
Replikat aktualisieren	– Aufsetzpunkt bestimmen mit Hilfe der Jobvariablen – Logging-Dateien bereitstellen – REFRESH REPLICATION	Hauptfunktion INFORMATION-SCHEMA Maske INF.5 Hauptfunktion COPY & RECOVER / REPLICATION (COP) Maske COP.7
Replikat um Anwender-Spaces erweitern	– Aufsetzpunkt bestimmen mit Hilfe der RECOVERY_UNITS – DA-LOG-Dateien bereitstellen – SESAM-Sicherung oder Fremdkopie bereitstellen – REFRESH SPACE	Hauptfunktion INFORMATION-SCHEMA Maske INF.5 Hauptfunktion COPY & RECOVER / REPLICATION (COP) Maske COP.7
Anwender-Spaces, Catalog-Space oder die gesamte Datenbank mit Hilfe eines Replikats reparieren, zurücksetzen oder neu erzeugen	– Aufsetzpunkt bestimmen mit Hilfe der Jobvariablen – Logging-Dateien bereitstellen – RECOVER USING/TO REPLICATION	Hauptfunktion INFORMATION-SCHEMA Maske INF.5 Hauptfunktion COPY & RECOVER / REPLICATION (COP) Masken COP.2.n.3 (n=1,2,4,5)

Tabelle 68: Utility-Anweisungen und Hauptfunktion des Utility-Monitors für das Anwenden eines Replikats

Eine detaillierte Beschreibung zum Arbeiten mit dem Replikat einer Datenbank finden Sie im Abschnitt „Replikat einer Datenbank“ auf Seite 247.

9.9 Fremdkopien einer Datenbank einsetzen

Eine Fremdkopie ist eine mit beliebigen Mitteln erstellte Kopie von Anwender-Spaces, eines Catalog-Space oder einer gesamten Datenbank. Mit SESAM/SQL können Fremdkopien erzeugt werden, ohne die Datenbank aus dem laufenden Betrieb zu nehmen.

In den folgenden Abschnitten wird beschrieben:

- welche Voraussetzungen zum Erzeugen einer Fremdkopie notwendig sind
- wie eine Fremdkopie erzeugt werden kann
- wofür eine Fremdkopie verwendet werden kann

9.9.1 Voraussetzungen für eine Fremdkopie

Bevor von einzelnen Spaces oder einer gesamten Datenbank eine Fremdkopie erzeugt werden kann, müssen Sie als SESAM/SQL-Systemverwalter die Spaces bzw. die Datenbank im DBH logisch oder physikalisch schließen.

ACHTUNG!
Wenn Sie einen Space kopieren, ohne ihn vorher logisch oder physikalisch geschlossen zu haben, erhalten Sie eine für ein RECOVER ungeeignete Sicherung. Bei einem RECOVER mit dieser Sicherung gibt SESAM/SQL eine Fehlermeldung aus.

Zum Schließen eines Space bzw. einer Datenbank können Sie eine der folgenden Maßnahmen wählen:

- die Spaces der Datenbank logisch schließen
  mit der Administrationsanweisung PREPARE-FOREIGN-COPY

  Beim logischen Schließen von Spaces einer Datenbank mit der Administrationsanweisung PREPARE-FOREIGN-COPY wird der Update auf die Spaces über eine Transaktionssperre unterbrochen und die Pufferinhalte werden in die Dateien geschrieben. Die Dateien bleiben aber weiterhin von SESAM/SQL geöffnet. Bis zur vollständigen Erzeugung der Fremdkopie darf kein Update erfolgen. Dies erreichen Sie, indem Sie die Administrationsanweisung innerhalb einer Locksequenz eingeben. Die Transaktionssperre gegen Update bleibt dann bis zum Ende der Locksequenz erhalten.

  Mit der Administrationsanweisung PREPARE-FOREIGN-COPY können Sie die gesamte Datenbank oder einzelne Anwender-Spaces logisch schließen. Die mit einer PREPARE-FOREIGN-COPY angesprochenen Spaces bilden ein Space-Set und haben den gleichen Sicherungszeitpunkt. Mit PREPARE-FOREIGN-COPY können Sie auch für die angegebenen Anwender-Spaces das Logging einschalten.

Wenn Sie die Datenbank nur logisch schließen, dann können Sie eine Fremdkopie nur mit Mitteln erzeugen, die offene Dateien sichern können. Hierzu zählt z.B. die Abspaltung einer Additional-Mirror-Unit. Dagegen ist das BS2000-Kommando COPY-FILE nur möglich, wenn die Datenbank bzw. der Space zuvor physikalisch geschlossen wurde.

- den Space physikalisch schließen mit der Administrationsanweisung CLOSE-SPACE

  Mit CLOSE-SPACE wird ein Space zwar physikalisch geschlossen, aber nicht gegen weitere Zugriffe geschützt. Diesen Zugriffsschutz erreichen Sie, indem Sie die Administrationsanweisung innerhalb einer Locksequenz eingeben.

- die Datenbank physikalisch schließen mit der Administrationsanweisung SET-SQL-DB-CATALOG-STATUS... STATUS=FREE oder mit der Administrationsanweisung PREPARE-FOREIGN-COPY... CLOSE=YES

  Nach dem physikalischen Schließen der Datenbank sind auch Fremdkopien möglich, die erfordern, dass die Datenbankdateien physikalisch geschlossen sind (z.B. SNAPs).

- den DBH herunterfahren mit der Administrationsanweisung STOP-DBH

Die genaue Syntax und Funktion der einzelnen Anweisungen sind im Handbuch „Datenbankbetrieb“ ausführlich beschrieben.

Beim Schließen der Datenbank werden Aufsetzinformationen für das Nachfahren der Loggingdateien in jeden Space eingetragen. Die Aufsetzinformationen enthalten:

- beim Catalog-Space den CAT-LOG-Eintrag in der CAT-REC-Datei, bis zu dem Änderungen im Catalog-Space enthalten sind
- bei den Anwender-Spaces den jeweiligen DA-LOG-Eintrag in der Catalog-Tabelle DA_LOGS, bis zu dem Änderungen in den Spaces enthalten sind

Die Fremdkopie einer Datenbank muss aus der CAT-REC-Datei, dem Catalog-Space und den Anwender-Spaces bestehen.
Der Space, von dem die Fremdkopie erzeugt werden soll, darf sich nicht im Zustand „recover pending“, „check pending“ oder „load running“ befinden.

9.9.2 Erzeugen einer Fremdkopie

Eine Fremdkopie ist eine Kopie eines Space, eines Catalog-Space oder einer gesamten Datenbank, die nicht mit SESAM-Mitteln erstellt wurde. Im Folgenden wird beschrieben, wie Sie Fremdkopien erzeugen mit:

- den Replikationsfunktionen der Storage-Systeme in Verbindung mit SHC-OSD
- dem BS2000-Kommando COPY-FILE

9.9.2.1 Erzeugen einer Fremdkopie mit Replikationsfunktionen

Folgende Replikationsfunktionen der Storage-Systeme können zur Erstellung von Fremdkopien unter SESAM/SQL verwendet werden:

- TimeFinder/Mirror (Symmetrix):
  Vollständige Kopie der Daten auf Additional-Mirror-Units, auch **B**usiness **C**ontinuance **V**olume (BCV) genannt
- TimeFinder/Clone (Symmetrix):
  Unmittelbar verfügbare Kopie der Daten, als vollständige Kopie realisiert
  (mit `/ACTIVATE-CLONE ..., COPY-COMPLETE-DATA=*YES`)

  TimeFinder/Mirror und TimeFinder/Clone können auch in Verbindung mit SRDF verwendet werden.

  Das Produkt TimeFinder/Snap kann nicht zur Erstellung von Fremdkopien verwendet werden, da es keine komplette Kopie des Originals erstellt.
- SnapView Clone (CLARiiON CX) :
  Unmittelbar verfügbare Kopie der Daten, als vollständige Kopie realisiert
  (mit `/ACTIVATE-CLONE ..., COPY-COMPLETE-DATA=*YES`)

  Das Produkt SnapView Snap kann nicht zur Erstellung von Fremdkopien verwendet werden, da es keine komplette Kopie des Originals erstellt.
- Equivalent Copy (EC, ETERNUS DX):
  Unmittelbar verfügbare Kopie der Daten, als vollständige Kopie realisiert
  (mit `/ACTIVATE-CLONE ..., COPY-COMPLETE-DATA=*YES`)

  Das Produkt SnapOPC+ kann nicht zur Erstellung von Fremdkopien verwendet werden, da es keine komplette Kopie des Originals erstellt.

Fremdkopien von Spiegelplatten werden in folgenden allgemeinen Arbeitsschritten erstellt:

- Synchronisation von Original- und Spiegelplatte
- Trennung von Original- und Spiegelplatte
- Erstellen der Fremdkopie von der Spiegelplatte (getrennte Verarbeitung)
- ggf. Re-Synchronisation von Original- und Spiegelplatte

Die dafür benötigten Kommandos sind ausführlich im Handbuch „SHC-OSD (BS2000)“ beschrieben. Dort finden Sie auch Grafiken, die den Ablauf von Synchronisation und Trennung der Plattenpaare veranschaulichen.

Ablauf beim Erzeugen von Fremdkopien mit Replikationsfunktionen

Im Ablaufschema wird das Zusammenspiel von SESAM/SQL, SHC-OSD und BS2000 beim Erstellen von Fremdkopien dargestellt. Erläuterungen: siehe Seite 387.

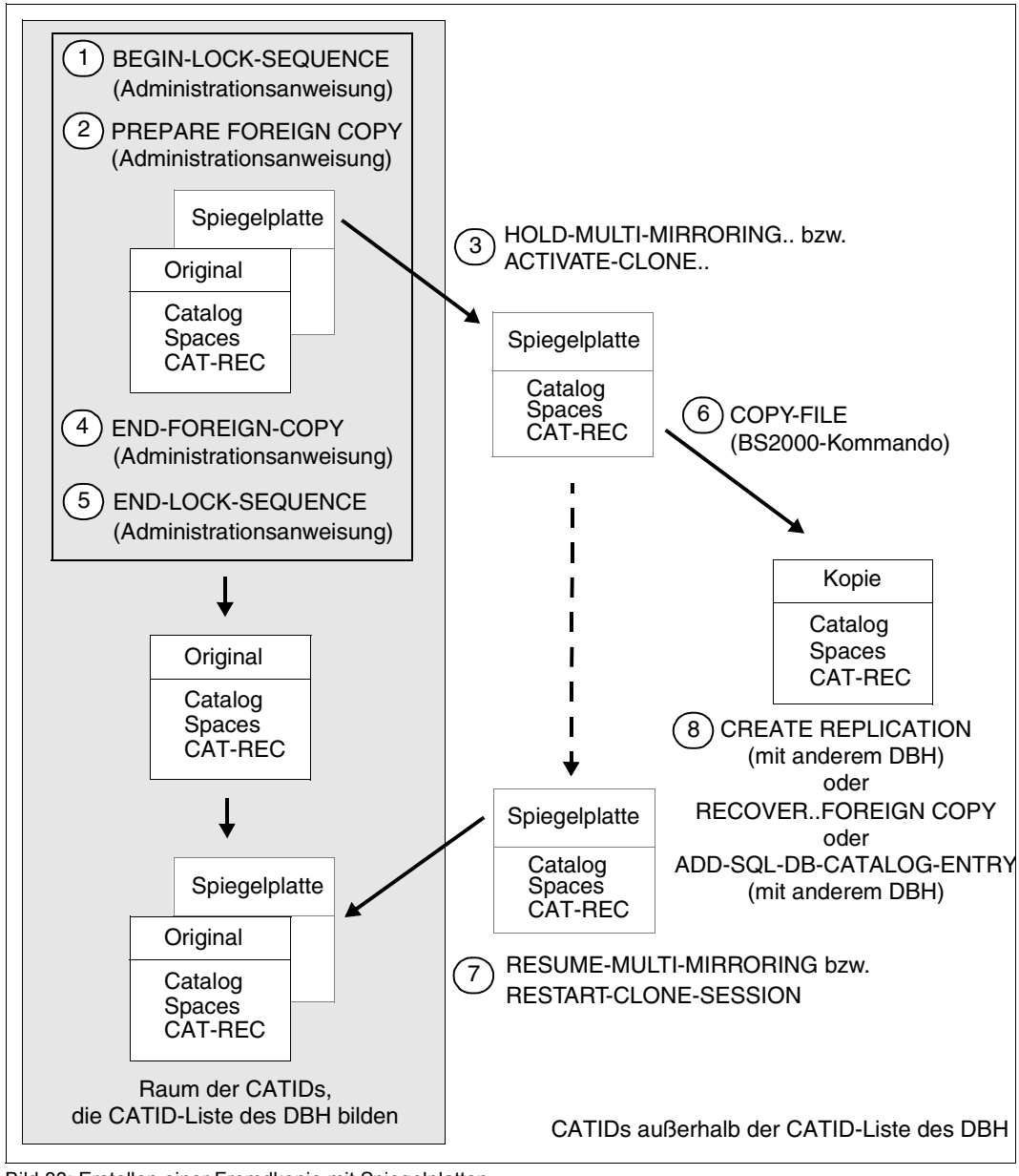


Bild 33: Erstellen einer Fremdkopie mit Spiegelplatten

1. Mit der Administrationsanweisung BEGIN-LOCK-SEQUENCE wird eine Locksequenz eröffnet.
2. Mit der Administrationsanweisung PREPARE-FOREIGN-COPY wird die Datenbank logisch geschlossen. Der Update auf dem Space wird dadurch über eine Transaktionssperre unterbrochen. Die Transaktionssperre gegen Update bleibt bis zum Ende der Locksequenz erhalten.
   Die Datenbank besteht aus der CAT-REC-Datei, dem Catalog-Space und den Anwender-Spaces.
3. Mit dem SHC-OSD-Kommando HOLD-MULTI-MIRRORING wird das Multimirror-Paar auf dem Symmetrix-System getrennt. Zur Trennung einer Clone-Unit in TimeFinder/Clone und SnapView/Clone dient das Kommando ACTIVATE-CLONE.

   Mit dem Parameter NEW-PUBSET wird die CATID der Spiegelplatte geändert. Die Spiegelplatte entspricht nun einer Kopie der Datenbank mit einer anderen CATID und gleichem Namen.

   - BS2000 verlangt eine Änderung der CATID, da die Spiegelplatte in dasselbe BS2000-System wie die Originaldatenbank importiert werden soll. Unverändert kann die Spiegelplatte nur in ein anderes BS2000-System importiert werden.
   - SESAM/SQL verlangt zusätzlich innerhalb der vom DBH betrachteten CATIDs eine Eindeutigkeit des Namens unabhängig von der CATID.
     Sie erreichen eine Eindeutigkeit des Namens, indem Sie der Spiegelplatte eine CATID zuweisen, die nicht in der CATID-Liste des DBHs ist, mit dem die Originaldatenbank angesprochen wird.

   Wenn Sie dem DBH keine CATID-Liste zugewiesen haben oder die Kopie auf einer CATID der CATID-Liste liegen soll, müssen Sie bei COPY-FILE (Schritt 6) den Namen der Kopie ändern.
4. Mit der Administrationsanweisung END-FOREIGN-COPY können Sie den Status „copy pending“ aufheben, das durch eine Utility-Anweisung entstanden ist, die der Fremdkopie vorausgegangen ist. Sie können auch den Datenbankstatus nach PREPARE-FOREIGN-COPY mit physikalischem Schließen der Datenbankdateien ändern.
5. Mit der Administrationsanweisung END-LOCK-SEQUENCE wird die Locksequenz beendet und damit auf der Originaldatenbank die Transaktionssperre gegen Update aufgehoben. Bei dem nächsten Zugriff auf die Originaldatenbank wird dieser wieder logisch geöffnet.

6. Mit dem BS2000-Kommando COPY-FILE wird eine Kopie der Spiegelplatte erzeugt. Bitte beachten Sie, dass SESAM/SQL innerhalb der vom DBH betrachteten CATIDs eine Eindeutigkeit des Namens unabhängig von der CATID verlangt.

   - Sie erreichen eine Eindeutigkeit des Namens, indem Sie der Kopie eine CATID zuweisen, die nicht in der CATID-Liste des DBHs ist, mit dem die Originaldatenbank angesprochen wird.
   - Beim COPY-FILE können Sie einen neuen Namen für die Kopie wählen.
     Der Name muss bei dem Kopiervorgang nur dann geändert werden, wenn Sie dem DBH keine CATID-Liste zugewiesen haben oder die Kopie auf einer CATID der CATID-Liste liegen soll.

7. Mit dem Kommando RESUME-MULTI-MIRRORING wird das Multimirror-Paar auf dem Symmetrix-System wieder rekonstruiert. Der erneute Start einer Clone-Session in TimeFinder/Clone und SnapView/Clone wird mit dem Kommando RESTART-CLONE-SESSION eingeleitet.

   Danach ist es möglich, eine neue Fremdkopie der Originaldatenbank zu erzeugen.

8. Um die Kopie der Spiegelplatte als Fremdkopie für SESAM/SQL-Anwendungen einsetzen zu können, ist je nach Verwendungszweck einer der folgenden Schritte notwendig:

   - CREATE REPLICATION
     Einem DBH weisen Sie eine CATID-Liste zu, die zwar die CATID der Kopie, aber nicht die CATID der Originaldatenbank enthält. Über diesen DBH wird nun mit der Utility-Anweisung CREATE REPLICATION ein Replikat der Originaldatenbank erzeugt.
   - RECOVER ... FOREIGN COPY
     Zunächst müssen Sie die Dateien der Fremdkopie selbst auf die entsprechenden Spaces der Originaldatenbank kopieren.
     Anschließend können Sie die Originaldatenbank mit der Utility-Anweisung RECOVER ... FOREIGN COPY reparieren oder zurücksetzen.
   - ADD-SQL-DB-CATALOG-ENTRY
     Einem DBH weisen Sie eine CATID-Liste zu, die zwar die CATID der Kopie, aber nicht die CATID der Originaldatenbank enthält. Im SQL-Datenbankverzeichnis dieses DBH tragen Sie die Fremdkopie ein als Duplikat der Originaldatenbank.

Wird die Kopie aus der Spiegelplatte mit Hilfe von HSMS erstellt, brauchen Sie die Punkte 3 - 7 nicht selber auszuführen. HSMS führt die erforderlichen Schritte durch, wenn Sie in der BACKUP-FILES-Anweisung den Parameter CONCURRENT-COPY= *YES(WORK-FILE-NAME=*BY-ADD-MIRROR-UNIT) angeben.

Physikalisch geöffnete Dateien, deren Konsistenz mit PREPARE-FOREIGN-COPY hergestellt worden ist, können mit HSMS gesichert werden.

Auch SESAM-Sicherungsbestände können mit Spiegelplatten erstellt werden, siehe „SESAM-Sicherungsbestand mit HSMS von Spiegelplatten erstellen“ auf Seite 370.

Beispielprozedur

In der folgenden Prozedur wird mittels SESADM eine Additional-Mirror-/Clone-Unit abgetrennt:

```
/ASSIGN-SYSDTA  TO-FILE=*SYSCMD
/START-SESADM
//START-DBH-ADMINISTRATION ...
//BEGIN-LOCK-SEQUENCE
//PREPARE-FOREIGN-COPY SELECT=*LOGICAL(CATALOG-NAME=...)
//HOLD-PROGRAM
/HOLD-MULTI-MIRRORING ... bzw. /ACTIVATE-CLONE ...
/RESUME-PROGRAM
//END-FOREIGN-COPY SELECT=*LOGICAL(CATALOG-NAME=...)
//END-LOCK-SEQUENCE
//END
```

Die Datenbankdateien auf der abgetrennten Spiegelplatte können jetzt gesichert werden.

9.9.2.2 Erzeugen einer Fremdkopie mit dem BS2000-Kommando COPY-FILE

Zum Erzeugen einer Fremdkopie einer gesamten Datenbank mit BS2000-Mitteln sind folgende Schritte erforderlich:

1. Mit der Administratoranweisung SET-SQL-DB-CATALOG-STATUS ... STATUS=FREE veranlasst der SESAM/SQL-Systemverwalter den DBH, die Originaldatenbank freizugeben (siehe Handbuch „Datenbankbetrieb“).
2. Mit dem Kommando COPY-FILE erstellt der Datenbankverwalter auf BS2000-Ebene Kopien aller Space-Dateien, d.h. des Catalog-Space, aller Anwender-Spaces und der CAT-REC-Datei. Die Dateinamen der Kopien müssen den Konventionen von SESAM/SQL genügen und lauten für:

   Catalog-Space :*catid*:*benutzerkennung*.*catalog*.CATALOG

   Anwender-Spaces :*catid*:*benutzerkennung*.*catalog*.*space*

   CAT-REC-Datei :*catid*:*benutzerkennung*.*catalog*.CAT-REC

catid	bezeichnet die BS2000-Katalogkennung.
benutzerkennung	bezeichnet die BS2000-Benutzerkennung, die für alle kopierten Spaces dieselbe sein muss.
catalog	bezeichnet den physikalischen Datenbanknamen des Datenbank-Duplikats.
space	bezeichnet den Namen der Kopie eines Anwender-Space. Für alle Anwender-Spaces muss *space* bei Original und Duplikat jeweils gleich sein.

3. Mit der Administrationsanweisung SET-SQL-DB-CATALOG-STATUS ... STATUS=ACTIVE trägt der Systemverwalter die Originaldatenbank wieder in das SQL-Datenbankverzeichnis der laufenden DBH-Session ein.

Einzelne Anwender-Space können gesichert werden, indem der Space mit CLOSE SPACE bzw. INTR CLOSE geschlossen und dann mit COPY-FILE gesichert wird. Um zu verhindern, dass SESAM/SQL den Space bei einem Zugriff sofort wieder öffnet, sollten die Anweisungen innerhalb einer Locksequenz eingegeben werden.

Mit den Administrationsanweisungen PREPARE-FOREIGN-COPY und END-FOREIGN-COPY kann ein „copy pending“ aufgehoben werden, das durch eine der Fremdkopie vorausgegangene Utility-Anweisung entstanden ist. Dazu setzen Sie vor Schritt 1 die Anweisung PREPARE-FOREIGN-COPY und nach Schritt 3 die Anweisung END-FOREIGN-COPY ab.

Beispielprozedur

In der folgenden Prozedur wird mit dem Kommando COPY-FILE eine Kopie der Datenbank erzeugt:

```
/ASSIGN-SYSDTA  TO-FILE=*SYSCMD
/START-SESADM
//START-DBH-ADMINISTRATION ...
//BEGIN-LOCK-SEQUENCE
//PREPARE-FOREIGN-COPY SELECT=*LOGICAL(CATALOG-NAME=...)
//HOLD-PROGRAM
/COPY-FILE ...
/RESUME-PROGRAM
//END-FOREIGN-COPY SELECT=*LOGICAL(CATALOG-NAME=...)
//END-LOCK-SEQUENCE
//END
```

Die kopierten Datenbankdateien können jetzt gesichert werden.

9.9.3 Replikate aus Fremdkopien erstellen

Aus Fremdkopien können Sie als Datenbankverwalter in gleicher Weise Replikate erzeugen wie aus SESAM-Sicherungsbeständen (siehe Abschnitt „Replikat einer Datenbank“ auf Seite 247). Mit einem Replikat können Sie:

- einen DBH im OLTP-Betrieb von rein lesenden Anwendungen entlasten
- eine Originaldatenbank reparieren
- eine Originaldatenbank zurücksetzen
- eine eigenständige Datenbank erzeugen

Wenn Sie ein Replikat oder Teilreplikat aus einer Fremdkopie erstellen, darf sich der Space beim Erzeugen der Fremdkopie nicht im Zustand „check pending“ oder „recover pending“ befinden.
Die Fremdkopie wird durch ihre CAT-REC-Datei spezifiziert.
Mit dem Parameter RENAME der Utility-Anweisung CREATE REPLICATION kann der Ablauf beschleunigt werden. Das Replikat entsteht dann nicht durch Kopieren der Fremdkopie, sondern durch Umbenennen. Dabei ist allerdings zu beachten, dass diese Anweisung nicht wiederholbar ist, auch nicht im Fehlerfall. Es muss eine neue Fremdkopie erstellt werden.

9.9.4 Reparieren und Zurücksetzen mit Fremdkopien

Zum Reparieren und Zurücksetzen mit RECOVER können auch Fremdkopien statt der mit COPY erzeugten SESAM-Sicherungsbestände eingesetzt werden. Mit Fremdkopien können folgende Sicherungseinheiten wiederhergestellt werden:

- einzelne Anwender-Spaces
- eine Space-Liste
- der Catalog-Space
- die gesamte Datenbank

Ein RECOVER auf Basis einer Fremdkopie erfolgt im Wesentlichen nach den gleichen Regeln wie auf Basis von SESAM-Sicherungsbeständen (siehe Abschnitt „Aufgaben im Rahmen des Media-Recovery“ auf Seite 368 und Abschnitt „Datenbank, Catalog-Space und Anwender-Spaces wiederherstellen“ auf Seite 231). Besonderheiten bei der Verwendung von Fremdkopien für ein RECOVER werden im Folgenden beschrieben.

Besonderheiten in der Verwendung von Fremdkopien bei RECOVER

Die Dateien der Fremdkopie muss der Anwender selbst auf die entsprechenden Spaces kopieren. Dazu müssen die Spaces geschlossen sein. SESAM/SQL fährt in diesem Fall nur die Loggingdateien auf Basis der in den Spaces vorhandenen Aufsetzinformationen nach. SESAM-Sicherungsbestände bleiben dabei unberücksichtigt.

RECOVER von Anwender-Spaces mit Fremdkopie

Sollen Anwender-Spaces mit Hilfe einer Fremdkopie repariert werden, müssen die Spaces beim Erzeugen dieser Fremdkopie im Logging gewesen sein. Der DA-LOG-Eintrag, der in der Fremdkopie als Aufsatzpunkt vermerkt ist, muss in den Metadaten noch eingetragen sein. Seit dem Erzeugen der Fremdkopie darf das Logging nicht unterbrochen worden sein. Spaces, die diese Bedingungen nicht erfüllen, bleiben unverändert. Auf sie kann später nur zugegriffen werden, wenn sie seit Erzeugen der Fremdkopie nicht geändert worden sind. Andernfalls wird bei einem Zugriffsversuch die Fehlermeldung ausgegeben, dass der Space inkonsistent ist.

Alle Anwender-Spaces, die gemeinsam bearbeitet werden (Angabe einer Space-Liste), müssen den gleichen Zeitstempel tragen (siehe Seite 382).

War ein Space nicht im Logging, als die Fremdkopie erzeugt wurde, kann er lediglich auf den Stand der Fremdkopie zurückgesetzt werden.

RECOVER des Catalog-Space mit Fremdkopie

Soll ein Catalog-Space mit Hilfe einer Fremdkopie repariert werden, muss der Aufsetzpunkt in der CAT-REC-Datei noch vorhanden sein. Es kann jeder Stand der CAT-REC-Datei verwendet werden, der bei oder nach dem Erzeugen der Fremdkopie des Catalog-Space entstanden ist.

Ist die CAT-REC-Datei Bestandteil einer Fremdkopie und wird sie für die Reparatur eingespielt, ist nur ein Zurücksetzen auf diese Fremdkopie möglich. In diesem Fall sind keine weiteren Einträge vorhanden, die nachgefahren werden könnten.

RECOVER einer Datenbank mit Fremdkopie

Bei der Reparatur der gesamten Datenbank repariert SESAM/SQL zunächst den Catalog-Space der Datenbank und anschließend die Anwender-Spaces.

Wird eine Datenbank mit Hilfe einer Fremdkopie repariert, bei der die Anwender-Spaces vor dem Catalog-Space gesichert wurden, dann werden gegebenenfalls Änderungen auf den Spaces nachgefahren. Wurden die Anwender-Spaces zu einem späteren Zeitpunkt als der Catalog-Space gesichert, dann ist ihre Aufsetzinformation in der DA_LOGS-Tabelle nicht vorhanden. Sie können nicht repariert werden.

War die gesamte Datenbank nicht im Logging als die Fremdkopie erzeugt wurde, so kann nur auf diesen Stand zurückgesetzt werden.

RECOVER einer Datenbank mit Fremdkopie auf einen Zeitpunkt

SESAM/SQL setzt den gesamten Catalog auf den Stand des angegebenen Zeitpunktes zurück. Wenn ANY nicht angegeben wird, dann muss *zeitstempel* den Zeitpunkt eines SESAM-Sicherungsbestandes des Catalog-Space bezeichnen. Der SESAM-Sicherungsbestand muss in der CAT-REC-Datei eingetragen sein.

SESAM verwendet die Fremdkopie des Catalog-Space und fährt die CAT-LOG-Dateien, die an diesen SESAM-Sicherungsbestand anschließen, bis zum angegebenen Zeitpunkt nach. Danach werden auf den Fremdkopien der Anwender-Spaces die DA-LOG-Dateien bis zum angegebenen Zeitpunkt nachgefahren.

9.9.5 Datenbank duplizieren

Von einer SESAM/SQL-Datenbank kann ein Duplikat erzeugt werden, d.h. eine voll einsatzfähige Datenbank, die hinsichtlich Aufbau und Inhalt mit der Originaldatenbank identisch ist. Um eine komplette Fremdkopie als Datenbank-Duplikat nutzen zu können, sind folgende Schritte notwendig:

1. Mit der Administrationsanweisung ADD-SQL-DB-CATALOG-ENTRY nimmt der Systemverwalter einen entsprechenden Eintrag ins SQL-Datenbankverzeichnis auf, damit auf das Datenbank-Duplikat zugegriffen werden kann. Dabei kann der Systemverwalter dem physikalischen Datenbanknamen einen beliebigen, bislang noch nicht vergebenen logischen Datenbanknamen zuordnen. Über diesen logischen Datenbanknamen kann das Duplikat angesprochen werden.
2. Falls für das Datenbank-Duplikat andere Speichermedien gelten als für die Originaldatenbank, passt der Datenbankverwalter die Definitionen von Storage Groups und Anwender-Spaces mit den SQL-Anweisungen ALTER STOGROUP und ALTER SPACE entsprechend an (siehe Handbuch „SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen“).
3. Falls die datenbank-spezifischen Dateien des Duplikats auf anderen Speichermedien angelegt werden sollen als die entsprechenden Dateien für die Originaldatenbank, passt der Datenbankverwalter die Medientabelle des Duplikats mit den Utility-Anweisungen ALTER MEDIA DESCRIPTION, CREATE MEDIA DESCRIPTION, DROP MEDIA DESCRIPTION entsprechend an (siehe Abschnitt „Ersten Mediensatz für DALOG- und PBI-Dateien in die Medientabelle aufnehmen“ auf Seite 349 sowie Abschnitt „Medientabelle pflegen“ auf Seite 349).
4. Mit der Utility-Anweisung COPY CATALOG erstellt der Datenbankverwalter einen SESAM-Sicherungsbestand des Duplikats (siehe Abschnitt „SESAM-Sicherungsbestand erstellen mit COPY“ auf Seite 368).
5. Der Datenbankverwalter löscht mit Hilfe des Utility-Monitors die Sätze in der CAT-REC-Datei, die vor der neuen Sicherung liegen und sich noch auf das Original beziehen.
6. Aus der Catalog-Tabelle RECOVERY_UNITS des Duplikats löscht der Datenbankverwalter mit der Utility-Anweisung MODIFY alle Informationen zu SESAM-Sicherungsbeständen von Anwender-Spaces, die sich noch auf das Original beziehen (siehe Abschnitt „Metadaten von SESAM-Sicherungsbeständen pflegen“ auf Seite 375).

ACHTUNG!
Soll das Datenbank-Duplikat auf einem anderen Server bearbeitet werden, müssen die Zugriffsrechte geändert werden, bevor das Duplikat erstellt wird. Hierzu werden in den Metadaten der Datenbank die Systemzugänge und Berechtigungen eingetragen, die einen Zugriff auf die Datenbank von dem anderen Server erlauben.

9.10 Platten-Reorganisation mit SPACEOPT

Mit dem kostenpflichtigen Softwareprodukt SPACEOPT werden Volumes eines Pubsets reorganisiert, um die im Laufe der Zeit entstandene Fragmentierung der Speicherbereiche zu beseitigen und die Anzahl der Extents der Dateien zu verringern.

SPACEOPT bereinigt eine Fragmentierung durch die optimale Verlagerung (Reorganisation) der Datei-Extents auf den Volumes eines Pubsets, siehe Handbuch „SPACEOPT (BS2000)“.

Mit SPACEOPT können auch geöffnete Dateien reorganisiert werden.
Folgendes ist aber zu beachten:

- Das Reorganisieren von Dateien ist mit einem physikalischen Verlagern von Datei-Extents verbunden. Deshalb sollte diese Funktion nur zu Zeiten durchgeführt werden, in denen die I/O-Last der betroffenen Volumes gering ist.
- Für das Reorganisieren der Dateien sperrt SPACEOPT temporär die entsprechenden BS2000-Katalogeinträge.
  Soll eine betroffene Datei geöffnet oder erweitert werden, so unterbricht SPACEOPT seine Verarbeitung, gibt seine Sperre auf, lässt den Auftrag passieren. SPACEOPT setzt dann nach erneutem Erwerb der Katalogsperre seine Verarbeitung fort. Dieser Vorgang wird pro Datei bis zu dreimal wiederholt, bevor die Datei von der Reorganisation ausgenommen wird.

10 Zusammenarbeit von SESAM/SQL und openUTM

SESAM/SQL-Server bildet zusammen mit dem universellen Transaktionsmonitor openUTM (BS2000) ein leistungsfähiges, vollintegriertes DB/DC-System für wiederanlauffähige Online Transaction Processing-Anwendungen.

Mit dem Produkt openUTM-D können SESAM/SQL-UTM-Anwendungen, die auf verschiedenen Rechnern liegen, Aufträge innerhalb von verteilten UTM-Transaktionen austauschen. Der Verarbeitungsauftrag wird zu den Rechnern weitergeleitet, wo sich die entsprechenden Programme und Datenbanken befinden.

openUTM-D übernimmt die Kommunikation mit den entfernten Systemen und überwacht die verteilten Transaktionen in Zusammenarbeit mit den Transaktionsmonitoren der verteilten Systeme. Beim Einsatz von openUTM-D arbeitet das Datenbanksystem entweder lokal oder auch verteilt, wenn zusätzlich SESAM/SQL-DCN eingesetzt wird.

10.1 UTM-Anwendungen

10.1.1 Architektur

In UTM-Anwendungen ist zwischen Anwenderprogramm und Konnektionsmodul noch das UTM-Anschlussmodul KDCROOT zwischengeschaltet (siehe Bild 34).

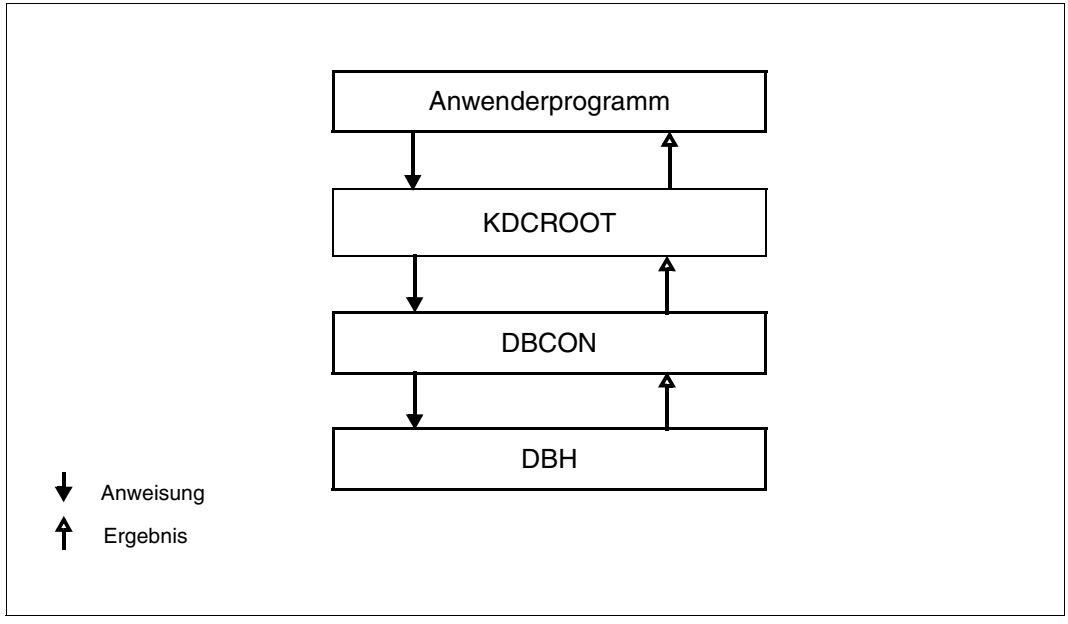


Bild 34: Datentransfer zwischen Anwenderprogramm und DBH

10.1.2 Erstellen des Anschlussprogramms KDCROOT

Bei der Generierung der SESAM/SQL-UTM-Anwendung muss angegeben werden, dass die UTM-Anwendung mit dem Datenbanksystem SESAM/SQL korrespondieren soll. Dies erfolgt mit der DATABASE-Anweisung von KDCDEF, siehe unten.

Der zum Übersetzen des KDCROOT-Tabellenmoduls erforderliche KDCDB-Makro steht in der SESAM/SQL-Makrobibliothek für UTM-Anwendungen. Die übrigen Makros sind in der UTM-Makrobibliothek enthalten. Für die Assemblierung müssen sowohl die Makros der UTM-Makrobibliothek als auch der KDCDB-Makro bereitgestellt werden.

KDCDEF-Anweisung DATABASE

DATABASE ENTRY=SESSQL,TYPE=SESAM [,LIB=*modlib*/LOGICAL-ID(SYSLNK)]	für SQL
DATABASE ENTRY=SESAM,TYPE=SESAM [,LIB=*modlib*/LOGICAL-ID(SYSLNK)]	für CALL-DML

LIB= Wenn der Parameter LIB= versorgt wird, dann wird das SESAM-Konnektionsmodul SESUTMC dynamisch nachgeladen.

modlib Name der BS2000-Bibliothek, die die SESAM-Module enthält. Damit wird das Konnektionsmodul aus dieser Modulbibliothek dynamisch nachgeladen. Ferner werden dann alle weiteren zum SESAM-Konnektionsmodul gehörenden Module aus dieser Bibliothek nachgeladen.

LOGICAL-ID(SYSLNK)
Das SESAM-Konnektionsmodul wird im IMON-Installationspfad für die SESAM/SQL-Modulbibliotheken gesucht und von dort aus nachgeladen.

Wenn mit SQL und CALL-DML gearbeitet werden soll, dann müssen beide Anweisungen angegeben werden, siehe auch „Beispiel 2“ auf Seite 400.

Empfehlungen

- Es wird empfohlen, das SESAM-Konnektionsmodul statisch zur UTM-Anwendung zu binden. In diesem Fall darf der Parameter LIB= in der KDCDEF-Anweisung DATABASE **nicht** angegeben werden. Das statische Binden hat den Vorteil, dass die generierte UTM-Anwendung nach einem SESAM-Versionswechsel nicht neu generiert werden muss.
- Wenn der SESAM-Konnektionsmodul nicht statisch zur UTM-Anwendung gebunden wird, dann sollte LIB=LOGICAL-ID(SYSLNK) angegeben werden. Dadurch wird die UTM-Anwendung bezüglich SESAM/SQL unabhängig von hardware- oder versionsabhängigen Installationspfaden bzw. Bibliotheksnamen. Voraussetzung ist eine ordnungsgemäße Installation mit IMON.

Beispiel 1

Das SESAM-Konnektionsmodul wird statisch gebunden, die Anwendung enthält keine CALL-DML-Teilprogramme. Dafür ist folgende DATABASE-Anweisung notwendig:

```
DATABASE ENTRY=SESSQL,TYPE=SESAM
```

Beispiel 2

Das SESAM-Konnektionsmodul wird statisch gebunden, in der Anwendung sind CALL-DML-Teilprogramme enthalten. In diesem Fall müssen zwei DATABASE-Anweisungen gegeben werden:

```
DATABASE ENTRY=SESSQL,TYPE=SESAM
DATABASE ENTRY=SESAM,TYPE=SESAM
```

Beispiel 3

Das SESAM-Konnektionsmodul wird dynamisch über den IMON-Installationspfad nachgeladen, in der Anwendung sind CALL-DML-Teilprogramme enthalten. Damit müssen zwei DATABASE-Anweisungen mit LIB-Parameter gegeben werden:

```
DATABASE ENTRY=SESSQL,TYPE=SESAM,LIB=LOGICAL-ID(SYSLNK)
DATABASE ENTRY=SESAM,TYPE=SESAM,LIB=LOGICAL-ID(SYSLNK)
```

Anschlussprogramm KDCROOT übersetzen

Vor der Übersetzung des Anschlussprogramms KDCROOT weisen Sie die SESAM-Makrobibliothek zu. Die Zuweisung ist vom verwendeten Assembler abhängig.

Beispiel 1

SESAM-Makrobibliothek zuweisen bei ASSEMBH:

```
/ADD-FILE-LINK LINK-NAME=UTMLIB,FILE-NAME=utm_maclib
/ADD-FILE-LINK LINK-NAME=SESAMLIB,FILE=sesam_maclib
/START-PROGRAM FROM-FILE=$ASSEMBH
//COMPILE  -
//SOURCE=kdcroot
//,MACRO-LIBRARY=(*LINK(LINK-NAME=SESAMLIB) -
.
.
```

Beispiel 2

SESAM-Makrobibliothek zuweisen mit ASSGEN:

```
/FILE sesam-maclib,LINK=ALTLIB2
/EXEC $ASSGEN
*COMOPT ALTLIB2
.
.
```

10.1.3 Systemzugang für SQL-Anwendungen

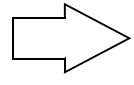
Ein UTM-Benutzer weist sich im System openUTM durch seine UTM-System-Benutzerkennung aus. Die UTM-System-Benutzerkennung setzt sich zusammen aus dem symbolischen Rechnernamen, dem Namen der UTM-Anwendung und dem KDCSIGN- bzw. LSES-Namen (bei UTM-D auf Auftragnehmerseite).

Um eine SESAM/SQL-Datenbank mit SQL bearbeiten zu können, benötigt ein Anwender einen Berechtigungsschlüssel eines SQL-Benutzers. Für diesen Berechtigungsschlüssel muss es einen entsprechenden Systemzugang in der Datenbank geben.

Beim Einrichten eines Systemzugangs wird einem bestehenden Berechtigungsschlüssel eine System-Benutzerkennung zugewiesen. Als System-Benutzerkennung können die Angaben für einen UTM-Benutzers oder einen BS2000-Benutzer eingetragen werden.

Ein SQL-Benutzer wird durch alle Systemzugänge im Datenbanksystem repräsentiert, die den Berechtigungsschlüssel dieses SQL-Benutzers enthalten.

Systemzugang	
Berechtigungsschlüssel	**System-Benutzerkennung**
UNIVUSR	HOST2/BS2USERID2
AUTHORIZE1	HOST1/BS2USERID1
AUTHORIZE1	HOST1/UTM1/KDC1
AUTHORIZE1	HOST1/BS2USERID3
AUTHORIZE2	HOST2/UTM2/KDC2



SQL-Benutzer

Bild 35: Beispiel für Systemzugänge eines SQL-Benutzers

Wie sich ein SQL-Benutzer mit dem Berechtigungsschlüssel gegenüber SESAM/SQL ausweist, ist im Abschnitt „Berechtigungsschlüssel eines SQL-Benutzers bekannt geben“ auf Seite 174 beschrieben.

10.1.4 Binden einer SESAM/SQL-UTM-Anwendung

Beim Binden wird aus dem Anschlussprogramm KDCROOT und den einzelnen Teilprogrammen die ablauffähige UTM-Anwendung erstellt.
Der Anschluss von SESAM/SQL an die UTM-Anwendung erfolgt über das Konnektionsmodul SESUTMC.

Das Konnektionsmodul SESUTMC muss zur UTM-Anwendung gebunden werden, wenn wie empfohlen bei der Generierung des KDCROOT in den DADATBASE-Anweisungen für SESAM/SQL keine Bibliothek angegeben wurde, siehe auch „Empfehlungen“ auf Seite 400.

Wird über die CALL-DML-Schnittstelle sortiert, so muss der Sortieranschlussmodul SESORT zur UTM-Anwendung gebunden werden.

Sollen Teilprogramme dynamisch von UTM nachgeladen werden, muss dies bei der Generierung der UTM-Anwendung angegeben werden (KDCDEF).

Das Binden einer UTM-Anwendung ist im openUTM-Handbuch „Anwendungen generieren und betreiben“ beschrieben.

10.1.5 Starten einer SESAM/SQL-UTM-Anwendung

Konnektionsmodul-Parameter

Zur Erfüllung seiner Funktionen benötigt DBCON verschiedene Parameter. Wichtige Konnektionsmodul-Parameter sind der DBH-Name und der Konfigurationsname. Diese Parameter ordnen das Anwenderprogramm, das mit dem Konnektionsmodul gebunden wird, einem DBH und einer Konfiguration zu. Daneben gibt es noch einige weitere Parameter zur Konfigurierung von DBCON.

Konnektionsmodule für UTM-Anwendungen, die mit dem independent DBH zusammenarbeiten (DBCON), werden normalerweise über die Konfigurationsdatei des Anwenderprogramms parametrisiert (siehe Abschnitt „Konfigurationsdatei“ auf Seite 296).

Der Anwender trägt die Konnektionsmodul-Parameter in die Konfigurationsdatei ein und weist diese vor dem Starten des mit dem Konnektionsmodul gebundenen Anwenderprogramms zu. Im Laufe der Startprozedur werden die eingetragenen Parameter dann an das Konnektionsmodul übergeben.

Alle Parameter sind DBCON-Parameter.

Die Konnektionsmodul-Parameter, die für UTM-Anwendungen gelten, finden Sie hier in alphabetischer Reihenfolge:

CCSN=*ccs-name* Codierter Zeichensatz, mit dem das Anwenderprogramm arbeitet.

ccs-name: Name des codierten Zeichensatzes, so wie er im BS2000 (Systemkomponente XHCS) definiert ist bzw. *NONE, wenn für das Anwenderprogramm kein codierter Zeichensatz angegeben werden soll. Standardwert: *NONE

Wenn für die Datenbank ein codierter Zeichensatz angegeben wurde (Klausel CODE-TABLE *ccs_name* in den Anweisungen CREATE CATALOG oder ALTER CATALOG), dann muss, bei Zugriffen des Anwenderprogrammes auf die Datenbank, der CCS-Name der Datenbank mit dem CCS-Namen des Anwenderprogrammes übereinstimmen. Zugriffe von Anwenderprogrammen mit CCSN=*NONE bzw. von Anwenderprogrammen aus SESAM/SQL < V5.0 werden mit SQLSTATE abgewiesen.

Für die Utility-Anweisungen CREATE/ALTER CATALOG, CREATE/REFRESH REPLICATION und RECOVER CATALOG [SPACE] findet diese Prüfung nicht statt.

Wenn für die Datenbank kein CCSN angegeben wurde, dann findet diese Prüfung ebenfalls nicht statt.

CNF=*k* Name der Konfiguration, in der das Anwenderprogramm arbeiten soll

k: A..Z, 0..9, ␣
Standardwert: ␣

DIAG-DUMP=(*diag*) Kriterium für die Erstellung eines Diagnose-Dump.

diag: {SQLSTATE=*state*|STATUS=*status*}

state: SQLSTATE mit Class und Subclass (5 Byte);
status: CALL-DML-Status mit Unternummer (4 Byte)

SQLSTATE kann teilqualifiziert angegeben werden durch Eingabe von „***“ als Subclass.
CALL-DML-Status kann teilqualifiziert angegeben werden durch Eingabe von „**“ als Unternummer.

Beim ersten Auftreten der entsprechenden Fehlermeldung (SQLSTATE bzw. CALL-DML-Status) wird ein Dump der Anwendertask erstellt (siehe Handbuch „Datenbankbetrieb“).

ISOL-LEVEL=*level* Standardwert für den Isolationslevel von SQL-Transaktionen. Nur wirksam in SQL-Anwendungen.

level: {READ-UNCOMMITTED | READ-COMMITTED | REPEATABLE-READ | SERIALIZABLE}
Standardwert: SERIALIZABLE

Gültig für alle Transaktionen des ESQL-Anwenderprogramms, wenn der Isolationslevel nicht durch die entsprechende SQL-Anweisung im Programm festgelegt ist.

Dieser Parameter wird nur ausgewertet von TIAM- und UTM-Anwendungen.

NAM=*x* Name des DBH, mit dem das Anwenderprogramm zusammenarbeiten soll

x: A..Z, 0..9, ␣
Standardwert: ␣

NOTYPE Meldung „SESAM*xx* not available ...“ wird unterdrückt

NOUNT Wenn eine langlaufende Suchfrage von SESAM/SQL unterbrochen wurde, wird kein Status an das CALL-DML-Teilprogramm gemeldet.

Wenn weder UNT noch NOUNT angegeben wird, ist NOUNT Standard.
Nur relevant für CALL-DML-Anweisungen!

NVT Die Anwendung kann nur lokal arbeiten und nur mit dem DBH kommunizieren, der beim Parameter NAM eingetragen wird. Dies gilt auch, wenn SESDCN geladen ist.

PREFETCH-BUFFER=*puffergröße*
Größe des Speicherbereichs, der für den Prefetch verwendet wird, in Kbyte.
Wertebereich: $0 \leq puffergröße \leq 4096$

Dieser Parameter wird nur von TIAM- und UTM-Anwendungen ausgewertet.

PRIO-CHECK=*t* Zeitintervall zwischen zwei aufeinanderfolgenden Prüfungen der BS2000-Task-Priorität des Anwenderprogramms in Sekunden.
$10 \leq t \leq 3600$
Standardwert: 300

PRIO-CHECK=OFF Prüfung der BS2000-Task-Priorität des Anwenderprogramms wird nach der ersten Prüfung abgeschaltet.

PUF=*p*

maximale Nachrichtenlänge bei DCAM- bzw. UTM-Anwendungen in Byte
$1 \leq p \leq 64000$
Standardwert: 4096

Eine Nachricht kann sein:
– die Anforderung (Anweisung) eines Anwenderprogramms an den SESAM/SQL-DBH
– die Antwort des SESAM/SQL-DBH an das Anwenderprogramm

Da die Länge von SQL-Nachrichten nur schwer einschätzbar ist, empfiehlt es sich, bei Anwendungen mit SQL-Anweisungen für *p* den Wert 64000 zu wählen.

TRACE,TYPE=*type*

Protokollierung des Call- bzw. Message-Trace ab Start des Anwenderprogramms einschalten.

type:

{CALL|MSG | (CALL,MSG)}
[,OUTPUT={SYSOUT | SYSLST | (SYSOUT,SYSLST)}]
Standardwert für Ausgabe (OUTPUT): SYSLST

Weitere Informationen zum Call- und Message-Trace siehe Handbuch „Datenbankbetrieb“.

TOTAL-APPL=*a*

maximale Anzahl SESAM/SQL-Teilhaber-Anwendungen in der Konfiguration (nur relevant für Anwendungen, die nicht verteilt arbeiten)
$1 \leq a \leq 128$
Standardwert: 64

TOTAL-USERS=*u*

maximale Anzahl SESAM/SQL-Teilhaber in der Konfiguration (nur relevant für Anwendungen, die nicht verteilt arbeiten)
$1 \leq u \leq 16000$
Standardwert: 128

UNT

Status 16 wird an das CALL-DML-Teilprogramm durchgereicht, wenn eine langlaufende Suchfrage von SESAM/SQL unterbrochen wurde.
Nur wirksam bei CALL-DML-Anweisungen!

UTMVG={JA | YES} Ein DB-Vorgang ist genau einem UTM-Vorgang zugeordnet.

Betriebsmittel werden nicht einem Anwender, sondern einem Vorgang zugeteilt. Sie können von einem nachfolgenden Vorgang nicht weiterverwendet werden. Am Ende des Vorgangs gibt SESAM/SQL die Betriebsmittel automatisch frei.

Im Rahmen der von openUTM angebotenen Vorgangskellerung können DB-Vorgänge uneingeschränkt gekellert werden.
Im Vorgangsexit sind Aufrufe an SESAM/SQL (SQL- oder CALL-DML-Anweisungen) nicht erlaubt.

UTMVG=JA muss gesetzt sein, wenn bereits im ersten Teil eines UTM-Anmeldevorgangs Datenbankaufrufe eingegeben werden sollen.

UTMVG={NEIN | NO} Ein DB-Vorgang kann sich über mehrere UTM-Vorgänge erstrecken.

Die Betriebsmittel werden einem Anwender zugeteilt und keinem Vorgang. DB-Vorgänge dürfen daher nicht gekellert werden.
SQL-Betriebsmittel sind an den UTM-Vorgang gebunden und werden bei Vorgangsende automatisch freigegeben.
CALL-DML-Betriebsmittel eines Dialog-Vorgangs können in einem nachfolgenden Dialog-Vorgang weiterverwendet werden.
Im Vorgangsexit sind CALL-DML-Aufrufe erlaubt. SQL-Aufrufe sind im Vorgangsexit grundsätzlich nicht erlaubt.
Im ersten Teil eines UTM-Anmeldevorgangs sind Datenbankaufrufe nicht erlaubt.
Falls ein zweiter SESAM/SQL-UTM-Vorgang eröffnet wird, obwohl noch ein SESAM/SQL-UTM-Vorgang offen ist, gibt openUTM eine entsprechende Fehlermeldung aus.

Standardwert ist UTMVG=NEIN.

In einem CALL-DML-Anwenderprogramm können spezielle Startparameter im Programm eingegeben werden. Bei openUTM-Anwendungen sind das nur UNT und NOUNT (siehe Handbuch „CALL-DML Anwendungen“), NAM, NOTYPE und TRACE sind nicht erlaubt.

Startparameter der Anwendung

Die UTM-Anwendung benötigt beim Laden Startparameter für UTM, SESAM/SQL und evtl. das Formatierungssystem (FHS).

Die UTM-Startparameter werden in folgender Form eingegeben:

[.UTM] START startparameter[,...]

Die UTM-Startparameter sind im openUTM-Handbuch „Anwendungen generieren und betreiben“ beschrieben.

Die SESAM/SQL-Startparameter werden über die Konfigurationsdatei eingegeben. Wenn eine leere Konfigurationsdatei zugewiesen ist, wird der Konnektionsmodul mit Standardwerten parametrisiert.

Wenn das Formatierungssystem FHS verwendet wird, sind dafür Startparameter in folgendem Format anzugeben:

.FHS startparameter

Die Schreibweise und die Bedeutung dieser Startparameter sind im Handbuch „FHS (BS2000)“ beschrieben.

Beispiel: UTM-Anwendung mit ESQL-COBOL starten

Vor dem Starten einer UTM-Anwendung mit ESQL-COBOL-Teilprogrammen müssen Sie die SESAM/SQL-Modulbibliothek zuweisen. Es empfiehlt sich, benötigte Laufzeitsysteme beim Starten der UTM-Anwendung dynamisch nachzuladen.

Erweitern Sie die Startprozedur der UTM-Anwendung um folgende Anweisungen:

```
/ADD-FILE-LINK LINK-NAME=SESAMOML, FILE-NAME=sesam-modlib ———————— (1)
.
.
/CONNECT-SESAM-CONFIGURATION TO-FILE=globale konfigurationsdatei, -
/                                    CONFIGURATION-LINK=linkname    (2)
```

oder

```
/ADD-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=dateiname ————————————— (3)
/START-PROGRAM FROM-FILE=*MODULE(LIBRARY=benutzerbibl,ELEMENT=elementname - — (4)
/                               ,RUN-MODE=ADVANCED -
/                               (ALTERNATE-LIBRARIES =YES) -
/                               )
.UTM utm-startparameter ——————————————————————————————————————— (5)
.FHS fhs-startparameter ——————————————————————————————————————— (6)
.UTM END ——————————————————————————————————————————————————————— (7)

.
.
```

(1) SESAM/SQL-Modulbibliothek zuweisen.

(2) Konfigurationsdatei zuweisen.

(3) Dynamischen Bindelader DBL zum Starten der UTM-Anwendung aufrufen.

(4) UTM-Startparameter angeben. Eine Beschreibung der einzelnen Startparameter finden Sie im openUTM-Handbuch „Anwendungen generieren und betreiben“.

(5) Startparameter für das Formatierungssystem FHS angeben. Eine Beschreibung der einzelnen Startparameter finden Sie im Handbuch „FHS (BS2000)“.

(6) Eingabe der UTM-Startparameter beenden.

10.2 Transaktionskonzept

Eine openUTM-Transaktion beginnt mit dem Starten eines Teilprogramms und endet mit dem Setzen eines Sicherungspunktes durch Beenden des Teilprogramms (siehe openUTM-Handbuch „Konzepte und Funktionen“).

Der Sicherungspunkt ist Aufsetzpunkt nach dem Zurücksetzen der nachfolgenden Transaktion.

Ausnahme:
Bei bestimmten Beendigungsarten, z.B. PEND KP beendet sich das Teilprogramm, setzt aber keinen Sicherungspunkt und beendet die Transaktion nicht.

In einer openUTM-Transaktion darf nur eine Datenbank-Transaktion (DB-Transaktion) ablaufen.

- Bei CALL-DML-Transaktionen wird die Beendigung durch eine ETA-Anweisung angestoßen. Nach dieser ETA-Anweisung darf keine weitere DB-Anweisung innerhalb der openUTM-Transaktion erfolgen.
  Bei openUTM-Anwendungen sind die geketteten CALL-DML-Anweisungen „Anweisung; ETA“ und „ETA; BTA“ nicht zulässig.
- Bei SQL-Transaktionen innerhalb von openUTM-Transaktionen ist weder COMMIT WORK noch ROLLBACK WORK zulässig. Das Ende der DB-Transaktion wird durch den PEND angestoßen.

Das Zurücksetzen einer SESAM/SQL-openUTM-Transaktion bedeutet, dass sowohl die DB-Transaktion als auch die openUTM-Transaktion zurückgesetzt wird.

Das DB/DC-System SESAM/SQL-openUTM synchronisiert das Ende der openUTM-Transaktion mit dem Ende der DB-Transaktion (siehe openUTM-Handbücher „Anwendungen generieren und betreiben“ und „Konzepte und Funktionen“).

Die Synchronisation läuft wie folgt ab:

- Bei CALL-DML-Transaktionen:
  Das Ende der DB-Transaktion im openUTM-Teilprogramm wird angestoßen.
  Die DB-Transaktion wird dann nicht geschlossen, da noch Fehler auftreten können, die einen Abschluss der openUTM-Transaktion verhindern.
  SESAM/SQL informiert openUTM über den Wunsch „Ende der DB-Transaktion“ und verzögert dieses Ende bis zum Ende der openUTM-Transaktion.
  Danach ist das openUTM-Teilprogramm noch tätig mit Aufgaben, die zur openUTM-Transaktion, aber nicht mehr zur DB-Transaktion gehören, wie z.B. Format- oder Druckausgaben.
- Bei SQL-Transaktionen:
  Das Ende der DB-Transaktion und das Ende der openUTM-Transaktion werden gleichzeitig angestoßen (durch den PEND).

Die DB/DC-Transaktion ist erfolgreich abgeschlossen, wenn die DB-Transaktion ordnungsgemäß beendet wurde und die Änderungen von UTM-Bereichen festgeschrieben wurden. Gesperrte Betriebsmittel werden nun freigegeben.

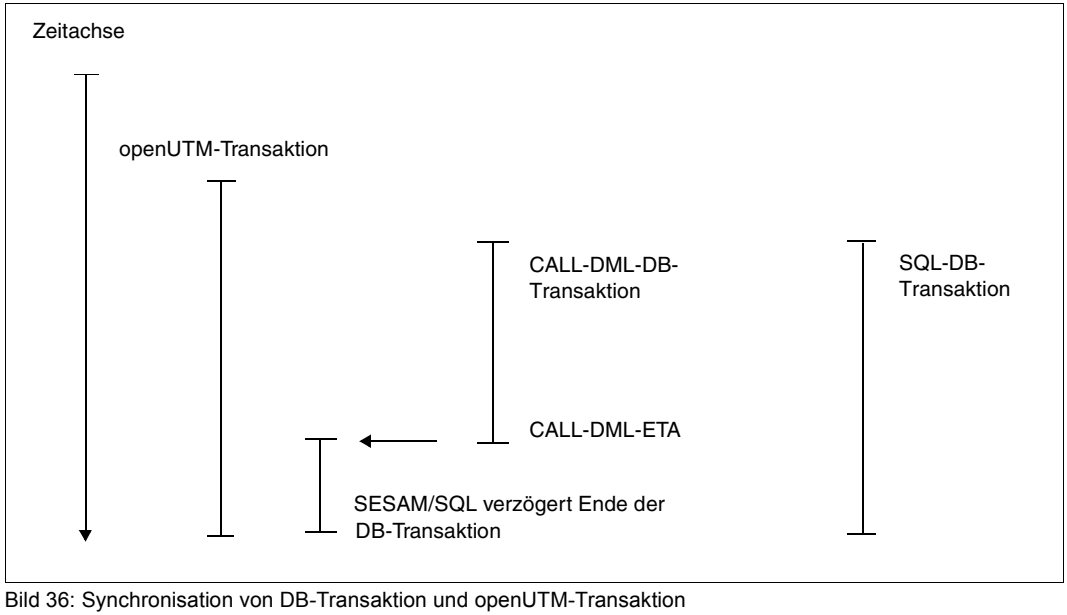


Bild 36: Synchronisation von DB-Transaktion und openUTM-Transaktion

10.3 Wiederanlauf

SESAM/SQL und openUTM führen einen synchronisierten Wiederanlauf durch.

SESAM/SQL benötigt dazu die Informationen über den Abarbeitungszustand der DB-Transaktionen in den Transaktionssicherungsdateien (TA-LOG1 und 2, WA-LOG, DDL-TA-LOG).
openUTM bezieht Informationen über offene bzw. abgeschlossene openUTM-Transaktionen aus der KDCFILE (Datei mit Daten für den Ablauf der openUTM-Anwendung).

Um zu gewährleisten, dass sowohl SESAM/SQL als auch openUTM alle für einen synchronisierten Wiederanlauf nötigen Informationen zur Verfügung stehen, soll der SESAM/SQL-DBH stets mit der Administrationsanweisung STOP-DBH UTM-SESSION-INFO=*KEEP (Standard) beendet werden (siehe Handbuch „Datenbankbetrieb“):
Durch STOP-DBH UTM-SESSION-INFO=*DELETE werden alle offenen Transaktionen zurückgesetzt, alle logischen Dateien geschlossen und die Wiederanlauf-Informationen gelöscht. Der Parameter UTM-SESSION-INFO=*KEEP bewirkt, dass die WA-LOG-Datei aktiv bleibt und die in ihr vermerkten Wiederanlauf-Informationen erhalten bleiben. Offene Transaktionen werden auch zurückgesetzt und logische Dateien geschlossen.

openUTM-Kalt- und openUTM-Warmstart

- Eine openUTM-Anwendung, die mit KDCSHUT normal beendet wurde, führt bei einem erneuten Starten der Anwendung einen *Kaltstart* durch (siehe openUTM-Handbuch „Anwendungen generieren und betreiben“).
- Kam es zu keinem ordnungsgemäßen KDCSHUT, wird beim erneuten Starten der Anwendung ein *Warmstart* durchgeführt.
  Der Warmstart ist nur dann erfolgreich, wenn vorher der DBH geladen wurde. In der Synchronisationsphase muss der DBH Auskunft geben können über die nicht beendeten Transaktionen, die sich beim Abbruch im „PEND“ befanden. Deshalb darf der DBH nicht zwischenzeitlich mit STOP-DBH UTM-SESSION-INFO=*DELETE beendet worden sein.

SESAM/SQL und openUTM führen abhängig vom Ende der vorherigen Session einen Kalt- bzw. Warmstart durch:

Ende der vorherigen Session		**Wirkung**	**Start der Session**
SESAM/SQL	/STOP-DBH UTM-SESSION-INFO=*DELETE	DBH setzt in der WA-LOG-Datei den Vermerk „nicht aktiv“, Auskunftsinformationen werden gelöscht	Kaltstart
openUTM	KDCSHUT N	KDCFILE enthält Wiederanlaufinformationen [1]	Kaltstart
SESAM/SQL	/STOP-DBH UTM-SESSION-INFO=*DELETE	DBH setzt in der WA-LOG-Datei den Vermerk „nicht aktiv“, Auskunftsinformationen werden gelöscht	Kaltstart [2]
openUTM	abgebrochen mit Fehler	KDCFILE enthält Wiederanlaufinformationen [3]	Warmstart [2]
SESAM/SQL	/STOP-DBH UTM-SESSION-INFO=*KEEP	WA-LOG-Datei bleibt aktiv, Auskunftsinformationen werden aktualisiert	Warmstart
openUTM	abgebrochen mit Fehler	KDCFILE enthält Wiederanlaufinformationen [3]	Warmstart
SESAM/SQL	abgebrochen mit Fehler	WA-LOG-Datei bleibt aktiv, Auskunftsinformationen bleiben erhalten	Warmstart
openUTM	abgebrochen mit Fehler	KDCFILE enthält Wiederanlaufinformationen [3]	Warmstart

Tabelle 69: Kalt- und Warmstart einer SESAM/SQL-openUTM-Anwendung

[1] KDCFILE kann Wiederanlauf-Informationen über offene Vorgänge, aber keine Informationen über nicht abgeschlossene Transaktionen enthalten.

[2] Normalerweise nicht ablauffähig, weil der SESAM/SQL-DBH einen Status Request nicht beantworten kann.

[3] KDCFILE kann sowohl Wiederanlauf-Informationen über offene Vorgänge als auch Informationen über nicht abgeschlossene Transaktionen enthalten

11 Anhang

11.1 Dateien und Jobvariablen von SESAM/SQL

In diesem Abschnitt sind alle Namen zusammengefasst, die SESAM/SQL benutzt, um die BS2000-Dateien und Jobvariablen zu bearbeiten.

Die in den folgenden Tabellen benutzten Variablen haben folgende Bedeutung:

k	Konfigurationsname (ein Byte)
n	DBH-Name (ein Byte)
st	Service-Task-ID (zwei Byte, von 01 bis 64)
ssss	TSN (vierstellig)
iiii	Startwert (vierstellig)
##(*##*)	laufende Nummer (zwei- oder vierstellig)
version	laufende Nummer der Kopie (sechsstellig)
catid	Katalogkennung
userid	beliebige Benutzerkennung
dcnid	Benutzerkennung der DCN-Task
catalog	Name des Catalog
space	Name des Space
replikat	Name des Replikats
jjjj.mm.tthh:mm:ss	Zeitstempel

11.1.1 Dateien von SESAM/SQL

In der folgenden Tabelle sind die sessionbezogenen Dateien aufgelistet, die vom SESAM/SQL-DBH erstellt werden.

Standardname der Datei	Bedeutung
:*catid*:$*userid*.SESAM.CO-LOG.*ssss*.*iiii*	Auftrags-Protokollierung
:*catid*:$*userid*.SESAM*kn*.CURSOR.####	Cursor-Dateien
:*catid*:$*userid*.SESLK*kn*.CURSOR.####	Cursor-Dateien beim Einsatz eines linked-in DBH
:*catid*:$*userid*.SESAM*kn*.TA-LOG1 :*catid*:$*userid*.SESAM*kn*.TA-LOG2	Transaktionssicherungsdateien
:*catid*:$*userid*.SESLK*kn*.TA-LOG1 :*catid*:$*userid*.SESLK*kn*.TA-LOG2	Transaktionssicherungsdateien beim Einsatz eines linked-in DBH
:*catid*:$*userid*.SESAM*kn*.WA-LOG	Logging-Datei für Wiederanlauf
:*catid*:$*userid*.SESLK*kn*.WA-LOG	Logging-Datei für Wiederanlauf beim Einsatz eines linked-in DBH

Tabelle 70: Sessionbezogene Dateien

Dateien, die von einem SESAM/SQL-DBH angelegt werden, der in einer Service-Task abläuft, haben zusätzlich im Standardnamen die Service-Task-ID, z.B.:
:*catid*:$*userid*.SESLK*kn**st***.TA-LOG1

In der folgenden Tabelle sind die datenbankbezogenen Dateien aufgelistet, die vom SESAM/SQL-DBH erstellt werden.

Standardname der Datei	Bedeutung
:*catid*:$*userid.catalog*.CATALOG :*catid*:$*userid.catalog.space*	Spaces einer Datenbank
:*catid*:$*userid.catalog*.CAT-REC	CAT-REC-Datei
:*catid*:$*userid.catalog*.CAT-REC.COPY	Kopie der CAT-REC-Datei
:*catid*:$*userid.replikat*.CAT-REC.REPL	CAT-REC-Datei eines Replikats
:*catid*:$*userid.catalog*.CATALG.*version* :*catid*:$*userid.catalog.space.version*	Sicherungen der Spaces
:*catid*:$*userid.catalog.jjjjmmtthhmmss.##*	PBI-Datei bzw. HSMS-Arbeitsdatei
:*catid*:$*userid.catalog.version*.C.####	CAT-LOG-Datei für Catalog Space
:*catid*:$*userid.catalog.version*.D.####	DA-LOG-Datei für Anwender-Space
:*catid*:$*userid.catalog.space*.EXC.L	Fehlerdatei für Utility-Anweisung LOAD
:*catid*:$*userid.catalog.space*.EXC.U	Fehlerdatei für Utility-Anweisung UNLOAD OFFLINE
:*catid*:$*userid.catalog.space*.EXC.C	Fehlerdatei für Utility-Anweisung CHECK FORMAL
:*catid*:$*userid.catalog.space*.DDLTA	Transaktionssicherungsdatei für DDL

Tabelle 71: Datenbank-bezogene Dateien

Die folgende Tabelle zeigt die Datei, die vom SESDCN erstellt wird.

Standardname der Datei	Bedeutung
catid.dcn-id.SES.DLG*k*	Sicherungsdatei von SESDCN

Tabelle 72: SESDCN-bezogene Datei

In der folgenden Tabelle sind die Dateien des Utility-Monitors dargestellt.

Standardname der Datei	Bedeutung
:*catid*:$*userid*.SESUTI.INPUTLOG*jjjjmmtthhmmss*	Ausgabedatei für Anweisungen
:*catid*:$*userid*.SESUTI.STDLOG*jjjjmmtthhmmss*	Default-Protokolldatei

Tabelle 73: Dateien des Utility-Monitors

Die folgende Tabelle enthält eine Übersicht über Dateien für die Bandsicherung.

Standardname der Datei	Bedeutung
:*catid*:$*userid.archive-directory-datei.* ARC-PAR	ARCHIVE-Parameterdatei
:*catid*:$*userid*.SESAM*kn*.ARC-PAR	ARCHIVE-Parameterdatei
:*catid*:$*userid.hsms-archiv-name*.HSMS-PAR	HSMS-Parameterdatei
:*catid*:$*userid*.SESAM*kn*.HSMS-PAR	HSMS-Parameterdatei

Tabelle 74: Dateien des Utility-Monitors

Die folgende Tabelle enthält eine Übersicht über Dateien für Diagnosezwecke.

Standardname der Datei	Bedeutung
:*catid*:$*userid*.SESAM*kn*.SYSLST	SYSLST-Protokoll der DBH-Task
:*catid*:$*userid*.SESAM*kn*.SYSOUT	SYSOUT-Protokoll der DBH-Task
:*catid*:$*userid*.SESAM*kn*.SYSLST.SESST80	SYSLST-Protokoll der Service-Task
:*catid*:$*userid*.SESAM*kn*.SYSOUT.SESST80	SYSOUT-Protokoll der Service-Task

Tabelle 75: Dateien des Utility-Monitors

11.1.2 S-Variablen von SESAM/SQL

In der folgenden Tabelle sind die Standardnamen der S-Variablen aufgelistet, die von SESAM/SQL bearbeitet werden.

Standardname der S-Variablen	Bedeutung
S-Variablen des Administrationsprogramms SESADM:	
SESADM-RESULT	Letzte Antwort einer Administrationsanweisung (Meldungsschlüssel)

Tabelle 76: Standardnamen der S-Variablen

11.1.3 Jobvariablen von SESAM/SQL

In der folgenden Tabelle sind die Standardnamen der Jobvariablen aufgelistet, die von SESAM/SQL bearbeitet werden.

Standardname der Jobvariablen	**Bedeutung**
DBH-spezifische Jobvariablen:	
SESAM.SESDBH.*kn*	Zustand der DBH-Session
SICHERUNGSINFORMATION.*kn*	TA-LOG-Zustand
Datenbankspezifische Jobvariable bei Replikaten:	
SESAM.*replikat*.NEXT-REPL-LOG [1]	Beinhaltet die älteste CAT-LOG- und DA-LOG-Datei, die zum nächsten REFRESH REPLICATION benötigt werden
Jobvariablen des Utility Monitors:	
SESAM.SESUTI.JV	Zustand des Utility-Monitors
#SESAM.SESUTI.JV	Zustand eines Utility-Monitors (parallele Überwachung mehrerer Utility-Monitore, die auf der selben Anlage und Benutzerkennung in verschiedenen Tasks ablaufen)
#SESAM.RU.CATALOG	Sicherungseinheit Catalog-Space
#SESAM.RU.*space*	Sicherungseinheiten Anwender-Spaces
#SESAM.RU.CAT-LOG	Logging-Dateien der Sicherungseinheit Catalog-Space
#SESAM.RU.DA-LOG	Logging-Dateien der Sicherungseinheiten Anwender-Spaces
Jobvariablen des Administrationsprogramms SESADM:	
#SESAM.SESADM.JV	Letzte Antwort einer Administrationsanweisung (Meldungsschlüssel)

Tabelle 77: Standardnamen der Jobvariablen

[1] Das Layout dieser Jobvariablen hat sich gegenüber SESAM/SQL V3.2 durch die jetzt 6-stellige Versionsnummer des SESAM-Sicherungsbestandes inkompatibel geändert

Alle Jobvariablen von SESAM/SQL sind im Modul SEZTXT in der SESAM/SQL-Modulbibliothek definiert. Die dazu gehörende Quelle SEZTXT.ASS wird als Bestandteil der Bibliothek SIPANY.SESAM-SQL.<ver>.SPEZ ausgeliefert. Der Anwender kann damit die Standardnamen der Jobvariablen ändern, siehe Handbuch „Datenbankbetrieb“, Abschnitt „Jobvariablen“.

11.2 Maximalgrößen von SESAM/SQL

In diesem Abschnitt sind alle Maximalgrößen aufgelistet, die in SESAM/SQL zu beachten sind.

11.2.1 Maximalgrößen für Basistabellen

Maximalgröße einer unpartitionierten Basistabelle	4 TB
Maximalgröße einer Partition	4 TB
Maximalgröße einer partitionierten Basistabelle	64 TB
Anzahl Partitionen einer Tabelle	16
Anzahl Sätze in einer Partition einer partitionierten Tabelle	268.435.454
Anzahl Sätze einer partitionierten Tabelle	4.294.967.264
Anzahl Sätze in einer nicht-partitionierten Tabelle	4.294.967.294
Anzahl Spalten in einer Tabelle mit Datentyp ungleich (N)VARCHAR	26.134
Anzahl Spalten in einer Tabelle mit Datentyp (N)VARCHAR	1.000
maximale Dimension einer multiplen Spalte	255
Anzahl Tabellen bzw. Partitionen pro Space	30.000
Anzahl Indizes pro Space	32.767

Tabelle 78: Maximalgrößen für Basistabellen

11.2.2 Maximalgrößen für Datenbankdateien

Die folgende Tabelle gibt einen Überblick über die Datenbankdateien und die datenbankspezifischen Dateien mit ihren Maximalgrößen.

Datei	**Maximalgröße**
Catalog-Space	4 TB auf Pubsets mit „großen Dateien“, 64 GB sonst
Anwender-Space	
SESAM-Sicherungsbestand auf Platte	
Arbeitsdatei auf Platte bei REORG	
CAT-REC-Datei, CAT-REC-Kopie, CAT-REC.REPL	64 GB
CAT-LOG- und DA-LOG-Datei	
DDL-TA-LOG-Datei	
PBI-Datei	
Arbeitsdatei auf Band bei REORG	Maximale Größe, die von der aktuellen BS2000-Version unterstützt wird
Arbeitsdatei (CRSI-Datei) bei CREATE INDEX und IMPORT TABLE	
Eingabedatei bei LOAD, Ausgabedatei bei UNLOAD	
Exportdatei bei EXPORT/IMPORT TABLE	
Fehlerdatei bei LOAD, UNLOAD, CHECK FORMAL oder ALTER TABLE	

Tabelle 79: Maximalgrößen für Datenbankdateien

11.2.3 Maximalwerte für das Arbeiten mit dem SESAM/SQL-DBH

Über DBH-Startanweisungen und -Optionen haben Sie die Möglichkeit, Grenzwerte für bestimmte Parameter festzulegen. Die folgende Tabelle zeigt eine Übersicht der Maximalwerte, die pro DBH-Session für wichtige Parameter gelten.

Maximalwert	**Anzahl**
Einträge im SQL-Datenbankverzeichnis	254
Einträge im CALL-DML-Tabellenverzeichnis	254
gleichzeitig zugreifbare Spaces	101 600
parallele Auftraggeber	32 767
parallele SQL-Zugriffspläne im Planpuffer	999 999
parallele SQL-Cursor	999 999
parallel belegte Suborders (Scans bei SQL bzw. logische Dateien bei CALL-DML)	262 143
DBH-Tasks	16
Service-Tasks beim independent DBH	64
CO-LOG-Dateien	9 999
sortierte Sätze in einer SQL-Cursor-Tabelle	2 147 483 647
Spaces pro Datenbank	1000
Threads beim independent DBH	1024
Schreibthreads beim independent DBH	512
Maximalwert	**Größe**
Bereich für Wiedergewinnungsanweisungen (in Spalten)	1024
Cursor-Puffer (in Kbyte)	1 500 000
System-Data-Buffer (in Kbyte)	64 000 000
Transfer-Container (in Kbyte)	1 000 000
User-Data-Buffer (in Kbyte)	64 000 000
Work-Container (in Kbyte)	1 000 000

Tabelle 80: Maximalwerte für das Arbeiten mit dem SESAM/SQL-DBH

11.2.4 Maximalwerte für das Arbeiten mit SESAM/SQL-DCN

Die folgende Zusammenstellung gibt einen Überblick über die Maximalwerte.
Diese ergeben sich aus den Systemgrenzen, die bei der verteilten Verarbeitung mit SESAM/SQL-DCN gelten:

Maximalwerte	**Systemgrenzen**
Anzahl Anwendungen in einer Konfiguration	128
Nachrichtenlänge in Byte	64000
Anzahl Einträge in der Verteilregel	340
Anzahl DBHs innerhalb einer verteilten Transaktion	13
Anzahl DBHs, bei denen gleichzeitig Aufträge aus einer Anwendertask aktiv sein können	170
Anzahl DBHs, die pro Konfiguration für verteilt arbeitende Anwender ansprechbar sind	25
Anzahl fremder Konfigurationen, die in einer Konfiguration bekannt sein können	406
Anzahl SQL-Scans bzw. logischer Dateien, die ein Auftraggeber gleichzeitig bearbeiten kann	ca. USERS * 0,75[1]

Tabelle 81: Maximalwerte für das Arbeiten mit SESAM/SQL-DCN

[1] USERS = zugelassene Anzahl Auftraggeber in der SESDCN-Session (DCN-Option)

Aufgrund der dynamischen Speicherplatzverwaltung ist es in Ausnahmefällen möglich, dass die maximale Anzahl SQL-Scans bzw. logischer Dateien, die ein Auftraggeber gleichzeitig bearbeiten kann, nicht erreicht wird.

11.2.5 Maximalwerte für Datentypen

Datentyp	Wertebereich
Alphanumerische und National-Datentypen (Länge in Zeichen)	
CHAR	$1 \leq länge \leq 256$
NCHAR	$1 \leq länge \leq 128$
VARCHAR	$1 \leq länge \leq 32.000$
NVARCHAR	$1 \leq länge \leq 16.000$
Numerische Datentypen (Wertebereich)	
SMALLINT	$-2^{15} \leq wert \leq 2^{15}-1$
INTEGER	$-2^{31} \leq wert \leq 2^{31}-1$
NUMERIC	0 oder $10^{-bruchteil} \leq wert \leq 10^{stellen\text{-}bruchteil} - 10^{-bruchteil}$
DECIMAL	0 oder $10^{-bruchteil} \leq wert \leq 10^{stellen\text{-}bruchteil} - 10^{-bruchteil}$
REAL (21 Binärstellen)	0 oder $5{,}4E^{-79} \leq wert \leq 7{,}2E^{75}$
DOUBLE PRECISION (53 Binärstellen)	0 oder $5{,}4E^{-79} \leq wert \leq 7{,}2E^{75}$
FLOAT (21/53 Binärstellen)	0 oder $5{,}4E^{-79} \leq wert \leq 7{,}2E^{75}$
Zeit-Datentypen (Wertebereich)	
DATE	0001-01-01 ≤ *datum* ≤ 9999-12-31
TIME	00:00:00.000 ≤ *zeit* ≤ 23:59:61.999
TIMESTAMP	0001-01-01 ≤ *datum* ≤ 9999-12-31 und 00:00:00.000 ≤ *zeit* ≤ 23:59:61.999

Tabelle 82: Maximalwerte für Datentypen

11.3 Hinweise zur Migration

Datenbanken ab SESAM/SQL V2.x werden beim ersten Zugriff durch einen DBH von SESAM/SQL V9.0 automatisch migriert. Bei der Migration wird der CCSN einer Datenbank aus SESAM/SQL < V5.0 auf den Wert _NONE_ geändert. Spaces mit logischer Datensicherung werden durch die Migration in den Zustand „copy pending“ versetzt. Es wird empfohlen, einen SESAM-Sicherungsbestand mit COPY CATALOG OFFLINE zu erstellen.
Als erste Anweisung ist auch COPY CATALOG_SPACE OFFLINE möglich. Die Anwender-Spaces können dann später mit COPY SPACE gesichert werden.

- Replikate können nicht migriert werden. Sie müssen neu erstellt werden.
- SESAM-Sicherungsbestände können nicht migriert werden. Sie müssen neu erstellt werden.
  SESAM-Sicherungsbestände aus SESAM/SQL ab V2.x können in SESAM/SQL V9.0 zum Lesen geöffnet werden. Datenbanken aus SESAM/SQL V1.x können nur mit der Utility-Anweisung MIGRATE migriert werden.
- Logging-Dateien aus SESAM/SQL V8.0 können in SESAM/SQL V9.0 nicht bearbeitet werden. RECOVER [USING] auf Basis einer Sicherung aus SESAM/SQL V8.0 ist deshalb in SESAM/SQL V9.0 nicht möglich.

  Möglich ist jedoch ein Rücksetzen auf eine Sicherung aus SESAM/SQL ab V2.x mit RECOVER SPACE ... TO. Dabei wird die Sicherung nicht migriert, sondern nach dem Einspielen wird der dabei entstandene Space nach SESAM/SQL V9.0 migriert. Der Space kommt in den Zustand „copy pending“ und muss gesichert werden.

  Möglich ist auch das Rücksetzen einer gesamten Datenbank auf eine Sicherung aus SESAM/SQL ab V3.1 in mehreren Schritten. Weil der Catalog-Space nach dem Rücksetzen zunächst den Zustand „copy pending“ hat, muss zuerst der Catalog-Space gesichert werden, bevor die Anwenderspaces zurückgesetzt werden können.

  1. Rücksetzen des Catalog-Space mit RECOVER CATALOG_SPACE ... TO. Dabei wird der Catalog-Space migriert.
  2. Sichern des Catalog-Space mit COPY CATALOG_SPACE.
  3. Rücksetzen der Anwender-Spaces mit RECOVER SPACESET unter Angabe des Zeitstempels der Catalog-Sicherung, auf die zurückgesetzt werden soll. Es können auch einzelne Spaces zurückgesetzt werden, wenn nur diese benötigt werden.
  4. Sichern der Spaces, die im Logging sind. Sie sind durch die Migration beim Rücksetzen in den Zustand „copy pending“ gekommen.

Eine Rückmigration aus SESAM/SQL V9.0 nach SESAM/SQL V8.0 ist möglich. Wenden Sie sich dazu bitte an Ihren Service-Beauftragten.

Ausführliche Informationen zum Thema „Migration“ finden Sie in der Freigabemitteilung zu SESAM/SQL V9.0.

Fachwörter

Dieses Fachwortverzeichnis enthält Definitionen wichtiger Begriffe, die in den Handbüchern zu SESAM/SQL verwendet werden.
Kursiv gedruckte Fachwörter in den erläuternden Texten verweisen auf entsprechende Definitionen für diese Fachwörter.
In der Zeile „Synonym(e):“ werden bedeutungsgleiche oder bedeutungsähnliche Bezeichnungen genannt, die in der Literatur, nicht jedoch in den SESAM/SQL-Handbüchern verwendet werden.
Ein „siehe“-Verweis für ein Fachwort verweist auf das in den SESAM/SQL-Handbüchern verwendete Fachwort.

In der Literatur zu relationalen Datenbanken werden häufig unterschiedliche Bezeichnungen synonym verwendet. So entspricht die Bezeichnung Relation des relationalen Modells der in der Praxis verwendeten Bezeichnung Tabelle, die Bezeichnung Tupel der Bezeichnung Zeile oder Satz, die Bezeichnung Attribut der Bezeichnung Spalte.
In den SESAM/SQL-Handbüchern werden die Bezeichnungen Tabelle, Satz und Spalte verwendet. Ausnahme: Das Handbuch „CALL-DML Anwendungen“ benutzt anstatt Spalte weiterhin die bis zur SESAM/SQL-V1.1 verwendete Bezeichnung Attribut.

Abfrage-Ausdruck
query expression

Teil einer *SQL*-Anweisung, die auf der Grundlage von *Basistabellen* oder *Views* eine neue *Tabelle* definiert, die sogenannte *Ergebnistabelle*. Ein Abfrage-Ausdruck ist z.B. ein *SELECT-Ausdruck*, ein *Join-Ausdruck* oder eine mit dem Schlüsselwort UNION verknüpfte Kombination von SELECT-Ausdrücken und Join-Ausdrücken.

Abrechnung
accounting

siehe *Accounting*

Abstrakte Tabelle
abstract table

Eine Tabelle, deren einzelne Zeilen nicht persistent gespeichert werden. Abstrakte Tabellen geben immer die momentan aktuellen Werte aus, die von SESAM/SQL aufgrund von Werten anderer Tabellen berechnet werden. In SESAM/SQL dienen abstrakte Tabellen als Grundlage für Views im Informationsschema.

Access-Handle
access handle

Der Begriff Access-Handle tritt im Zusammenhang mit Aufrufen des *SESAM-CLI* auf. Ein Access-Handle wird dann benötigt, wenn ein *BLOB-Wert* sequentiell gelesen oder geschrieben werden soll. Diese sequentielle Bearbeitung erfolgt in SESAM/SQL durch mehrmaliges Anwenden der entsprechenden Aufrufe des SESAM-CLI. Das Access-Handle verwaltet dabei die internen Informationen darüber, welcher BLOB-Wert aktuell bearbeitet wird und bis zu welchem Teilstück des BLOB-Werts die Bearbeitung fortgeschritten ist. Ein Access-Handle wird mit dem Aufruf SQL_BLOB_VAL_OPEN des SESAM-CLI erzeugt und mit dem Aufruf SQL_BLOB_VAL_CLOSE geschlossen.

Accounting
accounting

anwenderbezogenes Abrechnungsverfahren für alle Leistungen einer Session, wie etwa Anzahl der logischen und physikalischen Dateizugriffe.
Synonym: Abrechnung

Additional-Mirror-Unit
additional mirror unit

Zusätzliche *Spiegelplatte* in einem Symmetrix-Plattensystem, die ohne Beeinträchtigung des laufenden Betriebs für andere Zwecke (Sicherung, Testverarbeitung usw.) abgetrennt werden kann.
Synonym: Business Continuance Volume (BCV).
In anderer Literatur auch: BCV-Spiegel.

Administrationsanweisung
administration statement

Anweisung, mit der der *Systemverwalter* den *Data Base Handler (DBH)* und die *Verteilkomponente* SESDCN überwachen und steuern kann.
Die Eingabe von Administrationsanweisungen erfolgt über das Administrationsprogramm SESADM, das eine maskenorientierte Dialogoberfläche bietet.

Administrationskommando
administration command

Kommando, mit dem der *Systemverwalter* den *Data Base Handler (DBH)* und die *Verteilkomponente* SESDCN überwachen und steuern kann.
Die Eingabe von Administrationskommandos erfolgt - anders als bei *Administrationsanweisungen* - über das BS2000-Kommando INFORM-PROGRAM oder über eine *CALL-DML*-Anweisung innerhalb eines CALL-DML-Programms. Die Funktion von Administrationskommandos entspricht im Wesentlichen der Funktion von Administrationsanweisungen.

änderbar
updatable

Ein *View* oder *Cursor* heißt „änderbar“, wenn er zum Ändern der zugrundeliegenden *Basistabelle(n)* verwendet werden kann. Unabhängig von der Eigenschaft „änderbar“ eines Views oder Cursors müssen die *Privilegien* für die Basistabelle(n) entsprechend vergeben sein.

AES (Advanced Encrpytion Standard)

Ein Standard-Verschlüsselungsverfahren, das aus den USA stammt und heute überall verwendet wird. Es ist eine Chiffre, mit der Klartext in Blöcken fester Länge mit einem festen Algorithmus in Blöcke derselben Länge mit Geheimtext (Chiffretext) verschlüsselt wird. AES ist ein symmetrisches Verschlüsselungsverfahren, das denselben geheimen *Schlüssel* sowohl zum *Verschlüsseln* als auch zum *Entschlüsseln* verwendet.
In SESAM/SQL sind die Blöcke und der geheime *Schlüssel* jeweils 16 Byte (128 bit) lang (AES-128).

After Image
after image

Block nach erfolgter Datenänderung.

Aggregat
aggregate

Zusammenfassung von atomaren Werten. Innerhalb eines Datensatzes stellt ein Aggregat die Gesamtheit bzw. einen Teilbereich der Werte für eine *multiple Spalte* dar. In den Anweisungen INSERT und UPDATE werden Aggregate multiplen Spalten als Werte zugewiesen.

Annotation
annotation

Spezieller SQL-Kommentar. Gibt Hinweise für die Ausführung einer SQL- oder *Utility-Anweisung*. Eine Annotation wirkt sich, abhängig von ihrer Position, nur auf eine bestimmte Operation in der Anweisung aus.

Anwender-Space
user space

Space, in dem *Tabellen* und *Indizes* gespeichert sind. Anwender-Spaces sind zu unterscheiden vom *Catalog-Space*.

Anwenderdaten
user data

Nutzdaten in den *Anwender-Spaces* der *Datenbank*. Von den Anwenderdaten zu unterscheiden sind die *Metadaten* im *Catalog-Space* der Datenbank.

Anwendung
application

Umsetzung einer Aufgabenstellung in ein Anwenderprogramm (bzw. mehrere Anwenderprogramme), die mit SESAM/SQL-Datenbanken arbeiten.

Arbeitsgang
activity

bestimmte Funktionen des Utility-Monitors, in denen eine zusammenhängende Menge von Datenbankobjekten (Catalog, Schema, Tabelle) angelegt oder geändert wird.

Arbeitsleiste
internal statement format

optimiertes Format einer *CALL-DML*-Anweisung, das der *DBH* intern aus der betreffenden Anweisung des Anwenders erzeugt. Arbeitsleisten können bei Folgeanweisungen wiederverwendet werden.

Asynchronvorgang
asynchronous conversation

UTM-Vorgang, der vom *Auftraggeber* zeitlich entkoppelt abläuft. Asynchrone Aufträge bieten sich für Aufgaben und Nachrichten an, deren Rückmeldung der Auftraggeber für die weitere Bearbeitung nicht im nächsten Dialogschritt benötigt.

Attribut
attribute

siehe *Spalte*

Attributwert
attribute value

siehe *Spalte*

Auftraggeber
requesting user
: im *Teilnehmerbetrieb* ein Dialog- oder Batchprogramm. Im *Teilhaberbetrieb* ein Terminal (bei openUTM: UTM-Datenstation) bzw. ein Paar, bestehend aus Benutzerkennung und Terminal (bei openUTM: UTM-Benutzerkennung und beliebige UTM-Datenstation).

Ausdruck
value expression
: liefert einen numerischen oder alphanumerischen Wert oder einen Zeitwert. Ein Ausdruck kann eine *Spalte* einer Tabelle, eine *Konstante*, eine *Mengenfunktion*, eine *Benutzervariable*, eine *Unterabfrage* oder eine durch Operatoren verknüpfte Kombination dieser Elemente sein.

Ausprägung (einer multiplen Spalte)
occurrence
: siehe *multiple Spalte*

Autonome Transaktion
autonomous transaction
: Selbständige ablaufende *Transaktion* mit eigenen Thread und eigenem Transaktionskontext innerhalb einer umgebenden *Transaktion*.

Basistabelle
base table
: mit einer CREATE TABLE-Anweisung erstellte *Tabelle*, die permanent in einer *Datenbank* gespeichert wird. Im Rahmen der *Migration* werden ebenfalls Basistabellen erzeugt.
Basistabellen können von unterschiedlicher *Tabellenart* sein.
Basistabellen können partitioniert sein (*partitionierte Tabelle*).
Anzahl und *Datentyp* der Spalten sowie *Integritätsbedingungen* werden mit der *SQL*-Anweisung CREATE TABLE vereinbart und können mit ALTER TABLE modifiziert werden. Die Anzahl der *Zeilen* ist durch die Tabellendefinition nicht festgelegt.

Batchbetrieb
batch mode
: Betriebsart, in der ein Benutzerauftrag vollständig gestellt ist und von der Auftragstellung zeitlich entkoppelt abgewickelt werden kann. Die Betriebsart Batchbetrieb ist zu unterscheiden vom *Dialogbetrieb*.

Before Image
before image
: *Block* vor begonnener Datenänderung.

Benutzersicht
view

siehe *View*

Benutzervariable
host variable

Variable einer Wirtssprache (z.B. COBOL), die innerhalb einer *eingebetteten SQL*-Anweisung angesprochen wird. Benutzervariablen sind innerhalb von SQL-Anweisungen durch einen vorangestellten Doppelpunkt gekennzeichnet und müssen innerhalb der DECLARE-Section deklariert werden.

Berechtigungsschlüssel (eines SQL-Benutzers)
authorization identifier

Einem Berechtigungsschlüssel sind Privilegien zugeordnet, über die geregelt ist, welche Operationen (z.B. SELECT oder UPDATE) ein Benutzer auf die Datenbank ausführen darf. Der Berechtigungsschlüssel wird mit der *SQL*-Anweisung CREATE USER vergeben und mit CREATE SYSTEM_USER einer *System-Benutzerkennung* zugeordnet.

Big Endian

Eine Reihenfolge der Bytes einer Codierung im Speicher. Bei Big Endian liegt das höchstwertige Byte an der niedrigsten Speicheradresse. Fuer die *UTF-16 Code Units* (mit je 2 Bytes) wird von SESAM/SQL Big Endian verwendet. Antonym: *Little Endian*.

BLOB
BLOB (Binary Large Object)

BLOB ist ein Datentyp, mit dem in Datenbanken Multimedia-Dateninhalte gespeichert werden können, wie zum Beispiel Grafik, Video oder Sound.

BLOB-Objekt
BLOB object

In SESAM/SQL werden *BLOBs* als BLOB-Objekte verwendet, da sie nicht nur einen Wert sondern auch mehrere Eigenschaften besitzen.

BLOB-Tabelle
BLOB table

Besondere Basistabelle, in der nur *BLOB-Objekte* gespeichert werden können. Diese Tabelle wird mit der *SQL*-Anweisung CREATE TABLE ... OF BLOB erzeugt.

BLOB-Wert
BLOB value
: Der eigentliche Wert eines *BLOB-Objekts*. Er besteht aus einer Folge von Bytes mit variabler Länge, die bis zu 2^{31}-1 groß sein kann.

Block
block
: physikalische Dateneinheit von 4096 Byte, die bei SESAM/SQL die Zugriffseinheit darstellt.
Synonym: Datenblock

Blockfüllgrad
block utilization
: siehe *Freiplatzreservierung*

BS2000-System-Benutzerkennung
BS2000 system user ID
: siehe *System-Benutzerkennung*

Byte Order Mark
: Das Unicode Zeichen „zero-width no break space“, NX'FEFF'. Es wird manchmal als erstes Zeichen verwendet, um anzuzeigen, ob eine Zeichenfolge in der Codierungsform *UTF-16* im Format *Little Endian* oder *Big Endian* vorliegt.

CALL-DML
CALL DML
: Anweisungen einer speziellen *Data Manipulation Language (DML)*, deren Funktionen im Anwenderprogramm per Unterprogrammaufruf aktiviert werden. CALL-DML ermöglicht den transaktionsgesteuerten Datenbankzugriff.

CALL-DML-Modus
CALL DML mode
: Modus, in dem sich ein Anwenderprogramm befindet, wenn es gerade eine CALL-DML-Anweisung ausführt.

CALL-DML-Tabelle
CALL DML table
: siehe *Tabellenart*

CALL-DML-Tabellenverzeichnis
CALL DML table catalog list
: enthält einen Eintrag für jede *Tabelle* der *Tabellenart* CALL-DML-Tabelle, die während einer *DBH-Session* bearbeitet werden soll. Insbesondere wird dem *DBH* über das CALL-DML-Tabellenverzeichnis mitgeteilt, welche CALL-DML-

Tabelle welcher *Datenbank* zugeordnet ist. Für jede Datenbank, der eine CALL-DML-Tabelle zugeordnet ist, muss ein Eintrag im *SQL-Datenbankverzeichnis* existieren.
Der *Systemverwalter* richtet das CALL-DML-Tabellenverzeichnis per *DBH-Startanweisung* ein. Einträge in das CALL-DML-Tabellenverzeichnis kann der *Systemverwalter* über entsprechende *Administrationsanweisungen* hinzufügen oder löschen.

CAT-LOG-Datei
CAT LOG file

Datenbank-spezifische Datei. Auf CAT-LOG-Dateien werden Änderungen des *Catalog-Space* protokolliert (*CAT-Logging*). Die Medien für die CAT-LOG-Dateien werden über die *Medientabelle* verwaltet.

CAT-Logging
CAT logging

Sicherungsverfahren, bei dem sämtliche Änderungen des *Catalog-Space* auf einem Sicherungsmedium, der *CAT-LOG-Datei*, protokolliert werden. Zusammen mit dem *DA-Logging* ermöglicht das CAT-Logging im Rahmen des *Media-Recovery* die Reparatur einer defekten Datenbank bzw. eines defekten Space.

Catalog
catalog

benannte Zusammenfassung von *Schemata* einer *Datenbank*. Neben den anwenderdefinierten Schemata enthält der Catalog-Space der Datenbank stets die *Informationsschemata*. Über die Informationsschemata lassen sich die *Metadaten* der Datenbank abfragen.

Catalog-Recovery-Datei (CAT-REC-Datei)
catalog recovery file (CAT-REC file)

datenbank-spezifische Datei, die im Rahmen des *Media-Recovery* benötigt wird. Sie enthält Einträge zu Sicherungsbeständen (erstellt mit der *Utility-Anweisung* COPY) von *Catalog-Space*.
In der Catalog-Recovery-Datei ist jedem Sicherungsbestand eine laufende Nummer zugeordnet. Bei Reparatur bzw. Rücksetzen mit der Utility-Anweisung RECOVER wird ein Sicherungsbestand dann über die entsprechende Nummer identifiziert.
Beim COPY und bei Administration mit CHANGE-CATLOG wird eine CAT-REC-Kopie (*catalog*.CAT-REC.COPY) erzeugt und beim Erstellen eines Replikats eine CAT-REC-Datei des Replikats (*replikat*.CAT-REC.REPL).

Catalog-Space
catalog space

Der Catalog-Space enthält die Metadaten zu allen *Schemata* einer *Datenbank*, d.h. die Informationsschemata und anwenderdefinierte Schemata sowie weitere interne Verwaltungsinformation. Neben dem Catalog-Space gibt es die *Anwender-Spaces*.

CATID
CATID

siehe *Katalogkennung*

CATID-Liste
CATID list

Vom Anwender angegebene Liste von *CATID*s, die eine interne Suche nach Dateien auf darin angegebene *CATID*s beschränkt. Diese Liste kann dem DBH und dem Utility-Monitor beim Starten mit dem Linknamen SESAMCID zugewiesen werden.

Check-Bedingung
check constraint

Integritätsbedingung, die die zulässigen Datenwerte für eine oder mehrere *Spalten* einer *Tabelle* mit Hilfe einer *Suchbedingung* einschränkt. Ist eine Check-Bedingung vereinbart, werden bei den Anweisungen INSERT und UPDATE Sätze, die die Suchbedingung nicht erfüllen, nicht eingefügt bzw. nicht geändert.

CLI (Call Level Interface)
CLI

siehe *SESAM-CLI*

Client-Server-Architektur
client/server architecture

Architektur, bei der bestimmte Komponenten als sog. „Clients“ von einem oder mehreren anderen Komponenten, den sog. „Servern“, Dienstleistungen anfordern. Je nachdem, wie ausgeprägt die Rollenverteilung zwischen Server und Client ist, ergeben sich unterschiedliche Grundformen, z.B. entfernte Präsentation, entfernte Datenhaltung, verteilte Anwendung oder verteilte Datenbanken. Voraussetzung für die Realisierung einer Client-Server-Architektur sind geeignete Basis-Mechanismen für die Kommunikation zwischen gekoppelten Rechnern, die z.B. rechnerübergreifende *Transaktionssicherung* im Falle verteilter Transaktionsverarbeitung gewährleisten.

Code Point (Unicode Code Point)
code point

In Unicode wird jedem Zeichen eine Nummer, der so genannte Code Point, zugeordnet. In *SQL* kann ein Unicode Code Point in der Form U&'\xxxx' oder U&'\+xxxxxx' angegeben werden, wobei x eine hexadezimale Ziffer ist. Die Code Points liegen im Bereich von U&'\0000' bis U&'\10FFFF'.

Code Unit
code unit

Eine Code Unit ist die Einheit einer Unicode-Codierung. Z.B. ist eine Code Unit in *UTFE* 1 Byte (NX'nn'), in *UTF-16* 2 Byte (NX'nnnn') lang, wobei n eine hexadezimale Ziffer ist.

Codierter Zeichensatz (CCS)
coded character set (CCS)

Regeln, die die eindeutige Zuordnung von Zeichen eines Zeichensatzes mit ihrer Darstellung in Bits festlegen. Ein codierter Zeichensatz wird durch seinen Namen (*Codierter Zeichensatz Name*, CCSN) bezeichnet.

Codierter Zeichensatz Name (CCSN)
coded character set name (CCSN)

In einer SESAM/SQL-Datenbank gibt der CCSN (Klausel CODE_TABLE in den *SQL*-Anweisungen CREATE CATALOG bzw. ALTER CATALOG) an, mit welchem EBCDIC-Zeichensatz Werte, die in Spalten vom Datentyp [VAR]CHAR gespeichert sind, interpretiert werden.
Der CCSN einer SESAM/SQL-Anwendung (Parameter CCSN der Konfigurationsdatei) gibt an, mit welchem EBCDIC Zeichensatz die Anwendung Zeichenketten interpretiert.
Der CCSN eines Terminals (Parameter CODED-CHARACTER-SET beim BS2000-Kommando /MODIFY-TERMINAL-OPTIONS) gibt an, mit welchem EBCDIC-Zeichensatz Zeichen am Terminal dargestellt werden.
Der CCSN einer Datei (Parameter CODED-CHARACTER-SET beim BS2000-Kommando /MODIFY-FILE-ATTRIBUTES) gibt an, mit welchem Zeichensatz die Zeichen in der Datei zu interpretieren sind.

Collation
collation

Sortierreihenfolge, abhängig von der Codierung der Zeichen.
In der Unicode-Norm wird die Reihenfolge nach der die Unicode-Zeichen sortiert werden, wird mit Hilfe der Unicode Default Collation Table (DUCET) festgelegt. Diese Tabelle enthält eine Wertigkeit des Zeichens auf verschiedenen Ebenen.
In SESAM/SQL wird das Sortieren gemäß der Unicode Default Collation Table über die SQL-Funktion COLLATE() angeboten.

Collation-Element
collation element
Sortierelement der Unicode Default Collation Table (DUCET).
In SESAM/SQL das Ergebnis der SQL-Funktion COLLATE().

CO-LOG-Datei
CO LOG file
DBH-spezifische Datei für die Protokollierung von Aufträgen.
Über eine *Administrationsanweisung* kann der *Systemverwalter* die Auftragsprotokollierung aktivieren und den Datenträger für die CO-LOG-Datei auswählen.

Compound Index
compound index
siehe *Index*
Synonym: zusammengesetzter Index

Compound Key
compound key
siehe *zusammengesetzter Primärschlüssel*

COMPOUND-Anweisung
COMPOUND statement
Eine COMPOUND-Anweisung enthält *Prozeduranweisungen*, die in einem gemeinsamen Kontext ausgeführt werden. Fuer diese Prozeduranweisungen gelten gemeinsame *lokale Prozedurvariable*, gemeinsame *lokale Cursor* und gemeinsame *lokale Fehlerroutinen*, die alle innerhalb der COMPOUND-Anweisung deklariert werden. Eine COMPOUND-Anweisung darf keine weitere COMPOUND-Anweisung enthalten. COMPOUND-Anweisungen können nicht geschachtelt werden.

Constraint
constraint
siehe *Integritätsbedingung*

Consistency Check
consistency check
Die Komponenten von SESAM/SQL führen innerhalb ihres Aufgabenbereichs Konsistenzprüfungen durch. Auftretende Inkonsistenzen werden als Consistency Checks (zentrale Fehlermeldungen) auf Bedienstation bzw. Terminal und auf SYSLST gemeldet.

Cross Join
cross join

Join-Operation, deren *Ergebnistabelle* dem *Kartesischen Produkt* aus den beteiligten Tabellen entspricht.

CSV-Datei
CSV file

Standardisiertes Format für den plattform-unabhängigen Austausch von tabellarischen Daten (CSV: Comma Separated Values). Solche Dateien können mit vielen Softwareprodukten (z.B. mit SESAM/SQL oder Microsoft EXCEL) erzeugt werden.

Cursor
cursor

Zeiger innerhalb einer besonderen *Ergebnistabelle*, der sogenannten Cursortabelle, mit dem man auf die Sätze der Tabelle einzeln zugreifen kann. Bei der Deklaration des Cursors wird der Cursorname vergeben. In der Cursorbeschreibung wird die Cursortabelle spezifiziert und es wird festgelegt, ob der Cursor *änderbar* sein soll und die Sätze in der Cursortabelle eine bestimmte Reihenfolge haben sollen.

Cursor-Datei
cursor file

DBH-spezifische Datei, von der es zwei verschiedene Arten gibt:
- Datei, in die der *DBH* Zwischenergebnisse von Wiedergewinnungsanweisungen ablegt (interne Cursor-Datei)
- Datei, in der der Anwender Ergebnisse von CALL-DML-Suchfragen zwischenspeichern kann (Anwender-Cursor-Datei). Die Anwender-Cursor-Datei wird durch ein *Dateikennzeichen* identifiziert.

Cursor-Puffer
cursor buffer

Hauptspeicherbereich, den der *DBH* für die Zwischenergebnisse von Wiedergewinnungsanweisungen reserviert und verwaltet. Bei Pufferüberlauf lagert der DBH die Daten des Cursor-Puffers in eine oder mehrere interne *Cursor-Dateien* aus.

DA-LOG-Datei
DA LOG file

datenbank-spezifische Datei. Auf DA-LOG-Dateien werden alle DML-Änderungen (siehe *Data Manipulation Language*) in der *Datenbank* bzw. in einem *Space* protokolliert (*DA-Logging*).
Die Medien für die DA-LOG-Dateien werden über die *Medientabelle* verwaltet.

DA-Logging
DA logging
Sicherungsverfahren, bei dem sämtliche DML-Änderungen (siehe *Data Manipulation Language*) auf ein Sicherungsmedium, die *DA-LOG-Dateien*, protokolliert werden. Das DA-Logging kann entweder für die *Datenbank* insgesamt oder für bestimmte *Spaces* erfolgen. Zusammen mit dem *CAT-Logging* ermöglicht das DA-Logging im Rahmen des *Media-Recovery* die Reparatur einer defekten Datenbank bzw. eines defekten Space.

Data Base Handler (DBH)
database handler (DBH)
Komponente von SESAM/SQL, die alle Datenbankzugriffe einer *DBH-Session* analysiert, ausführt und koordiniert.
Der Data Base Handler (DBH) ist in zwei Varianten einsetzbar:
– independent DBH
Data Base Handler, der als selbständiges Programmsystem den *Mehrbenutzerbetrieb* unterstützt. Der independent DBH läuft unabhängig von der Anwenderprogramm-Task ab. Der DBH kann aus mehreren DBH- und Service-Tasks bestehen; die erste DBH-Task wird als Starttask, die weiteren als Folgetask bezeichnet.
– linked-in DBH
Data Base Handler, der exklusiv die Aufträge eines einzigen Anwenderprogramms bearbeitet und zum Anwenderprogramm gebunden wird. Der linked-in DBH läuft in derselben Task wie das Anwenderprogramm ab.
Synonym: SESAM/SQL-DBH, DBH

Data Base Management System
database management system
siehe *Datenbanksystem*
Abkürzung: DBMS

Data Definition Language (DDL)
data definition language (DDL)
bei Datenbanksprachen häufig gebrauchter Sammelbegriff für Anweisungen zur Definition von *Schemata*, *Tabellen*, *Privilegien* und *Integritätsbedingungen*. Die *SQL-Norm* fasst diese Anweisungen unter dem Begriff „SQL schema statements“ (*SQL*-Anweisungen zur Schemadefinition und -verwaltung) zusammen.

Data Manipulation Language (DML)
data manipulation language (DML)
bei Datenbanksprachen häufig gebrauchter Oberbegriff für Anweisungen zur Datenabfrage und -änderung. Die *SQL-Norm* verwendet den Begriff „Data Manipulation Language“ nicht. Stattdessen verwendet die SQL-Norm den Begriff „SQL data statements“ für *SQL*-Anweisungen zur Datenabfrage und -änderung im engeren Sinn. Daneben unterscheidet die SQL-Norm zur Klassifizierung der SQL-Anweisungen die Begriffe „SQL transaction statements“ (Anweisungen zur Transaktionsverwaltung), „SQL session statements“ (SQL-Anweisungen zur Sessionsteuerung“), „SQL dynamic data statements“ (Anweisungen der *dynamischen SQL*) sowie die WHENEVER-Anweisung zur ESQL-Fehlerbehandlung.

Dateikennzeichen (CALL-DML)
file identifier (CALL DML)
zweistelliges Kennzeichen zur Identifizierung einer *logischen Datei*, das beim Eröffnen der logischen Datei mit der CALL-DML-Anweisung Open vergeben wird. Es kann auch eine *Cursordatei* identifizieren.

Datenbank
database
zusammengehörige Datenbestände, die mit Hilfe eines *Datenbanksystems* ausgewertet, bearbeitet und verwaltet werden.
Bei SESAM/SQL besteht eine Datenbank aus den *Metadaten* im *Catalog-Space* und den Anwenderdaten in den zugehörigen *Anwender-Spaces*. Eine Datenbank wird durch den Datenbanknamen identifiziert.

datenbank-spezifische Datei
database-specific file
Datei, die pro Datenbank geführt wird und datenbank-spezifische Informationen enthält. Informationen über die datenbank-spezifischen Dateien sind in der *Medientabelle* hinterlegt.
DA-LOG-Datei, *CAT-LOG-Datei*, *Catalog-Recovery-Datei* und *PBI-Datei* sind datenbank-spezifische Dateien.

Datenbankadministration
database administration
siehe *Datenbankverwaltung*

Datenbankadministrator
database administrator
siehe *Datenbankverwalter*

Datenbankdatei
database file
BS2000-Datei, die einen *Space* realisiert. Die Datenbankdateien haben folgende Namen:
catalog.CATALOG für den *Catalog-Space*
catalog.space für beliebige Spaces
Dabei bezeichnet „catalog“ den bei CREATE CATALOG angegebenen Datenbank-Namen; „space“ bezeichnet den bei CREATE SPACE vergebenen Namen des betreffenden Spaces.

Datenbankkatalog
database catalog
siehe *CALL-DML-Tabellenverzeichnis*, *SQL-Datenbankverzeichnis*

Datenbanksystem
database system
Softwaresystem, das alle Aufgaben im Zusammenhang mit Verwaltung und Kontrolle großer Datenbestände unterstützt. Die im Datenbanksystem enthaltenen Verfahren führen zu einer stabilen, redundanzfreien und erweiterbaren Datenorganisation. Sie ermöglichen einer Vielzahl von Anwendern den parallelen Zugriff auf die *Datenbanken* und gewährleisten einen konsistenten Datenbestand.
Synonym: Data Base Management System

Datenbankverwalter
database administrator
für die *Datenbankverwaltung* zuständige Person bzw. Personengruppe.
Synonym: Datenbankadministrator

Datenbankverwaltung
database administration
Aufgabengebiet, das folgende Tätigkeiten umfasst:
– Generieren einer *Datenbank*
– Laden und Entladen von Daten
– Importieren und Exportieren von Tabellen
– Sichern und Wiederherstellen einer Datenbank bzw. von Teilen einer Datenbank; Verwalten der Sicherungsbestände.
– Reorganisieren und Prüfen der *Spaces*

Die einzelnen Tätigkeiten können aktiviert werden:
– innerhalb eines Anwenderprogramms mit *Utility-Anweisungen*
– über den *Utility-Monitor* (menügesteuert)

Synonym: Datenbankadministration

Datenblock
data block

siehe *Block*

Datenschutz
data protection

Schutz der Daten vor unberechtigtem Zugriff.
Datenschutz meint zum einen den Schutz des Einzelnen vor unbegrenzter Erhebung, Speicherung, Verwendung und Weitergabe seiner persönlichen Daten, wie er durch gesetzgeberische Maßnahmen und spezielle Datenschutzbeauftragte erreicht werden soll (juristischer Aspekt), zum andern alle Maßnahmen zur Realisierung dieses Schutzes.
Schutz vor unberechtigtem Zugriff auf *Datenbanken* wird zunächst durch einen entsprechenden *Systemzugang* erreicht, der bei SESAM/SQL durch einen *Berechtigungsschlüssel* und eine *System-Benutzerkennung* realisiert ist. In *SQL* erlauben das *View*-Konzept und die Vergabe von *Privilegien* es, verschiedenen Benutzern unterschiedlichen Zugriff auf die Objekte einer Datenbank zu gewähren.
Bei Tabellen der *Tabellenart* CALL-DML-Tabelle ist der Zugriffschutz über ein *Kennwort* geregelt. Außerdem können die Dateien des Datenbanksystems auf BS2000-Ebene durch ein BS2000-Passwort geschützt werden.

Datentyp
data type

Ein Datentyp legt den Bereich zulässiger Werte für ein Datenobjekt (z.B. *Spalte*) fest. In *SQL* wird der Datentyp einer Spalte mit den Anweisungen CREATE TABLE oder ALTER TABLE festgelegt. SESAM/SQL unterstützt numerische Datentypen (SMALLINT, INTEGER, NUMERIC, DECIMAL, REAL, FLOAT, DOUBLE PRECISION), alphanumerische Datentypen (CHARACTER, CHARACTER VARYING), Datentypen für Zeitwerte (DATE, TIME, TIMESTAMP) sowie Datentypen für *Vektoren*. Der *NULL-Wert* ist zulässiger Wert eines jeden Datentyps.

Datenwiedergewinnung
retrieval

lesender Zugriff auf die Daten in einer *Datenbank*.
Synonym: Wiedergewinnung

DB/DC-System
DB/DC system

Datenbanksystem (DB), das in Verbindung mit einem Datenkommunikationssystem (DC) sowohl die gemeinsame Nutzung von Verarbeitungsvorgängen als auch parallele Zugriffe auf gemeinsame Datenbestände ermöglicht. Dabei gewährleistet die gemeinsame, synchronisierte Datensicherung jederzeit einen konsistenten Zustand der Datenbanken und der Dateien des DB/DC-Systems. Der Einsatz von SESAM/SQL unter Steuerung des universellen Transaktionsmonitors openUTM bietet dem Anwender ein voll synchronisiertes DB/DC-System.

DB-Kennung
DB user ID

BS2000-Benutzerkennung ungleich der *DBH-Kennung*, in der die Datenbank, d.h. der *Catalog-Space*, seine CAT-REC-Datei und die *Anwender-Spaces* abgelegt sind. Die *Datenbank* kann in einer DB-Kennung oder der *DBH-Kennung* liegen.

DBH
DBH

siehe *Data Base Handler*

DBH-Kennung
DBH user ID

BS2000-Benutzerkennung, in der der SESAM/SQL-DBH als Datenbank-Server gestartet wurde.

DBH-Name
DBH name

identifiziert einen *DBH* eindeutig gegenüber anderen DBHs derselben *Konfiguration*. Auch einige *DBH-spezifische Dateien* werden durch den DBH-Namen identifiziert.
Der *Systemverwalter* vergibt den DBH-Namen mit einer *DBH-Option*.
Standardname: '␣'
Synonym: Kommunikationsname

DBH-Option
DBH option

DBH-Optionen sind Parameter für den *DBH*, die der *Systemverwalter* beim Starten des Betriebs angibt. DBH-Optionen legen für die aktuelle *DBH-Session* Grenzwerte, Betriebsmittel und Arbeitsregeln fest.

DBH-Session
DBH session

Zeitraum zwischen Starten und Beenden eines *DBH*.

DBH-spezifische Datei
DBH specific file

sessionbezogene Datei, die der *Data Base Handler (DBH)* anlegt.
Folgende Dateien sind DBH-spezifisch: *Cursor-Datei*, *TA-LOG-Dateien*, *WA-LOG-Datei*, *CO-LOG-Datei* und die Datei für die Protokollierung des DBH-Nachrichtenpuffers.
Informationen zu den Speichergeräten für Cursor-Datei, temporäre Arbeitsdateien, TA-LOG-Datei und WA-LOG-Datei sind im *Medienkatalog* hinterlegt.

DBH-Startanweisung
DBH start statement

Die DBH-Startanweisungen veranlassen das Einlesen der *DBH-Optionen* und das Einfügen von Einträgen in das *SQL-Datenbankverzeichnis* bzw. in das *CALL-DML-Tabellenverzeichnis*.
Der *Systemverwalter* gibt die DBH-Startanweisungen beim Starten des *DBH* an.

DBH-Task
DBH task

BS2000-Task, auf der die Basisfunktionen des DBH ablaufen. Beim *Multitasking* kann der DBH mit mehreren DBH-Tasks geladen werden. Darüber hinaus laufen bestimmte Teilfunktionen von SESAM/SQL auch in *Service-Tasks* ab.

DCN
DCN

siehe *SESAM/SQL-DCN (Verteilkomponente)*

DCN-Option
DCN option

DCN-Optionen sind Startparameter für die Verteilkomponente *SESAM/SQL-DCN*.

DDL-TA-LOG-Datei
DDL TA LOG file

Eine DDL-TA-LOG-Datei dient zur Absicherung von lang laufenden DDL- und SSL-Anweisungen, die in einer Service-Task ablaufen. Die DDL-TA-LOG-Datei entlastet damit die gewöhnlichen TA-LOG-Dateien. Sie ist dem betroffenen Space zugeordnet und wird nach dem Ende der Transaktion wieder gelöscht.

Deadlock
deadlock

Zustand, der eintritt, wenn sich zwei oder mehrere *Transaktionen* gegenseitig sperren.

Defaultwert
default value
siehe *SQL-Defaultwert*

Defaultwertzeichen
default value character
Zeichen, das bei *Tabellen* der *Tabellenart* CALL-DML/SQL-Tabelle für jede Spalte, die nicht im Primärschlüssel enthalten ist, gesondert vereinbart sein muss. Für jede solche Spalte ist dann ein *nicht-signifikanter Attributwert* definiert, der unter Verwendung des jeweils vereinbarten Defaultwertzeichens gebildet wird.

(SQL-)Deskriptorbereich
(SQL) descriptor area
Datenstruktur, mit der Ein- und Ausgabewerte für Anweisungen der *dynamischen SQL* beschrieben werden können. Jeder Wert und sein Datentyp wird durch einen Eintrag im Deskriptorbereich (item descriptor area) repräsentiert. Entsprechend existieren bei *multiplen Spalten* zu einem Wert (*Vektor*) mehrere Einträge.

Diakritisches Zeichen
diacritical mark
Mit einem Grundzeichen oder Symbol verknüpftes Zeichen, z. B. Akzent, Tilde.

Dialogbetrieb
interactive mode
Betriebsart, in der ein Benutzerauftrag als Folge einzelner Auftragsschritte gestellt wird, meist interaktiv von einem Datensichtgerät aus. Die Betriebsart Dialogbetrieb ist zu unterscheiden vom *Batchbetrieb*.

Direktänderung (CALL-DML)
direct update (CALL DML)
Datenänderung über die CALL-DML-Schnittstelle.

dirty read
dirty read
Phänomen, das in Abhängigkeit vom eingestellten *Isolationslevel* auftreten kann:
Eine *Transaktion* ändert einen *Satz* oder nimmt einen Satz neu auf. Eine zweite Transaktion ignoriert die *Satzsperre* und liest diesen Satz, bevor die erste Transaktion die Änderung festgeschrieben hat. Damit hat die zweite Transaktion einen Satz gelesen, der von der ersten Transaktion noch geändert oder gelöscht werden kann, also noch nicht den endgültigen Zustand hat.

Distributed Data Base Handler
distributed database handler
: die Gesamtheit aller *Data Base Handler*, die an der *verteilten Verarbeitung* teilnehmen.

dyadischer Operator
dyadic operator
: siehe *zweistelliger Operator*

dynamisch übersetzbare SQL-Anweisung
preparable SQL statement
: *SQL*-Anweisung, die erst zur Laufzeit des Programms in der Wirtssprache (z.B. COBOL) übersetzt wird. Auf diese Weise können beispielsweise Datenbankabfragen formuliert werden, die bei der Erstellung eines Programms noch nicht bekannt sind.

dynamische SQL
dynamic SQL
: bietet Anweisungen, um *dynamisch übersetzbare SQL*-Anweisungen zur Laufzeit einer SQL-Anwendung zu übersetzen und auszuführen, sowie zum Setzen und Lesen von *SQL-Deskriptorbereichen*. Mit der dynamischen SQL lassen sich z.B. Dialogprogramme mit variabel gestalteten Abfragen und Änderungen der Datenbank erstellen.

Eindeutigkeitsbedingung
UNIQUE constraint
: *Integritätsbedingung* für eine *Spalte* bzw. Spaltenkombination. Eine Eindeutigkeitsbedingung verhindert, dass zwei Sätze einer Tabelle für die angegebene Spalte(nkombination) dieselben Werte bzw. dieselben Wertekombinationen besitzen. Ist eine Eindeutigkeitsbedingung vereinbart, so sorgt SESAM/SQL dafür, dass in der angegebenen Spalte(nkombination) kein Wert bzw. keine Wertekombination doppelt vorkommt.

einfache Spalte
atomic column
: *Spalte*, für die im Gegensatz zu *multiplen Spalten* pro Datensatz nur ein einziger Wert gespeichert werden kann.

einfacher Primärschlüssel
single primary key
: *Primärschlüssel*, der aus einer einzigen *Spalte* besteht.

eingebettete SQL-Anweisung
embedded SQL statement

SQL-Anweisung, die innerhalb eines Programms der Wirtssprache (z.B. COBOL) angegeben wird und deren Anfang (mit EXEC SQL) und Ende (mit END-EXEC) markiert ist. Diese Markierungen heben die SQL-Anweisungen von den Anweisungen der Wirtssprache ab und ermöglichen die Precompilierung der SQL-Anweisungen.

einstelliger Operator
monadic operator

Operator mit nur einem Operanden. Die Vorzeichen + und - sind Beispiele für einstellige Operatoren.
Synonym: monadischer Operator

ECM (Electronic Codebook Mode)

Verfahren, bei dem mit einem Block-Verschlüsselungsverfahren (wie dem in SESAM/SQL verwendeten *AES*) ein Klartext verschlüsselt werden kann, der länger ist als ein Block. Im ECM wird der Klartext in einzelne Blöcke zerlegt und jeder Block mit demselben *Schlüssel* verschlüsselt. Der letzte Block muss ggf. verlängert werden.
In SESAM/SQL wird dieses Verfahren für Klartext verwendet, der länger als 16 Byte ist.
Wenn der Klartext eine Wiederholung einer Folge von Bytes enthält, so dass zwei der Blöcke mit Klartext exakt gleich sind, dann sind auch die entsprechenden Blöcke mit Geheimtext gleich. Solche Wiederholungen können also im verschlüsselten Text entdeckt werden, ohne den *Schlüssel* zu kennen.

Entschlüsselung
decryption

Die inverse Operation zur *Verschlüsselung*:
Die Umwandlung von Geheimtext in Klartext mit Hilfe eines Verschlüsselungs-Algorithmus und einem *Schlüssel*.
Wird in SESAM/SQL mit der SQL-Funktion DECRYPT() ausgeführt.

Entwertungszeichen
escape character

wird in der ESCAPE-Klausel eines LIKE-*Prädikats* definiert und muss im Vergleichswert dieses LIKE-Prädikats unmittelbar vor einem Zeichen „%“, „_“ oder einem weiteren Entwertungszeichen stehen. Dieses Zeichen „%“ oder „_“ bzw. das Entwertungszeichen verliert dadurch seine Funktion als Platzhalter bzw. Entwertungszeichen und wird als gewöhnliches Zeichen interpretiert.

Ergebnistabelle
derived table

Tabelle, die das Ergebnis eines *Abfrage-Ausdrucks* ist.

Export-Datei
export file

BS2000-Datei, in der Metadaten und Anwenderdaten einer Basistabelle beim Exportieren dieser Tabelle abgelegt werden. Über *SQL*-Anweisungen kann der Anwender auswählen, welche Anwenderdaten in eine neue Tabelle übertragen werden sollen. Die Export-Datei kann nicht händisch bearbeitet werden und wird ausschließlich zum Importieren einer Tabelle genutzt.

externer Wiederanlauf
external restart

Wiederanlauf nach Systemausfall, den der *DBH* durchführt, wenn er nach einer Abbruch-Session geladen wird.

Fehlerdatei
exception file

Datei, die der Anwender

– beim Ändern des *Datentyps* einer oder mehrerer *Spalten* einer *Basistabelle* angeben kann. Treten Konvertierungsfehler auf, schreibt SESAM/SQL die ursprünglichen Spaltenwerte zusammen mit der zugehörigen Fehlermeldung oder Warnung in die benannte Fehlerdatei.
– beim Laden einer Basistabelle angeben kann. Sie enthält Informationen über fehlerhafte Sätze der verwendeten Eingabedatei und die Ursache des Fehlers.

Datei, die von SESAM/SQL im Bedarfsfall bei CHECK FORMAL, LOAD oder UNLOAD angelegt wird. Sie enthält Informationen über fehlerhafte Datensätze und die Ursache des Fehlers. Für jede dieser drei Utility-Anweisungen legt SESAM/SQL eine eigene Fehlerdatei an, die mit jedem neuen Fehler fortgeschrieben wird.

Folgeänderung (CALL-DML)
follow-up update (CALL DML)

vereinfachte Form einer *Direktänderung*, die nach demselben Muster wie die vorhergehende Direktänderung arbeitet, jedoch andere *Attributwerte* verwendet.

Folgemaske
continuation form

Maske, in die der Utility-Monitor bei der Ausführung bestimmter Funktionen zur weiteren Verarbeitung verzweigt.

Freiplatzreservierung
free space reservation

gibt an, wieviel Prozent jedes *Blocks* nach der Ausführung der *Utility-Anweisungen* LOAD (Daten zuladen) und REORG (*Space* reorganisieren) frei bleiben sollen. Die Freiplatzreservierung wird durch den Parameter PCTFREE der Anweisungen CREATE CATALOG bzw. CREATE SPACE festgelegt. Entsprechend gibt der Blockfüllgrad den Prozentsatz eines Blocks an, der nach LOAD bzw. REORG maximal belegt ist.

Fremdkopie
foreign copy

Eine Kopie, die nicht mit der SESAM-Anweisung COPY erzeugt worden ist, sondern mit beliebigen BS2000-Mitteln. Auf Basis einer Fremdkopie kann eine *Recovery* durchgeführt oder ein *Replikat* erzeugt werden.

Fremdschlüssel
foreign key

Spalte einer *Tabelle*, die sich auf eine bestimmte Spalte mit der Eigenschaft PRIMARY KEY (siehe *Primärschlüsselbedingung*) oder UNIQUE (siehe *Eindeutigkeitsbedingung*) in einer anderen Tabelle bezieht.
Der Fremdschlüssel wird durch die *Referenzbedingung* definiert, die die Beziehung zwischen Fremdschlüssel und referenzierter Spalte herstellt. Die Tabelle, die die UNIQUE- oder die PRIMARY KEY-Spalte enthält, ist die referenzierte Tabelle, die Tabelle, die die Referenzbedingung enthält, ist die referenzierende Tabelle. Referenzbedingungen können auch zwischen Spaltenkombinationen bestehen.

Funktionskennzeichen (CALL-DML)
function identifier (CALL DML)

bestimmt für die Gültigkeitsdauer des jeweiligen *Dateikennzeichens* die zugelassenen Funktionen.

globale Konfigurationsdatei
global configuration file

Die globale Konfigurationsdatei fasst Konfigurationsdaten für mehrere Komponenten in einer Datei zusammen (siehe *Konfigurationsdatei*).

globale Transaktion
global transaction

Transaktion, die sich durch die Verteilung mittels SESAM/SQL-DCN über mehrere *Data Base Handler* erstreckt.

Group-Commit
group commit

Zusammenfassung mehrerer Aufträge „Ende Transaktion“ zu einem gemeinsamen Schreibauftrag.
Synonym: Sammelkonsistenzpunkt

Hauptfunktion
main function

abgeschlossene Arbeitseinheit, die ggf. die zusammenhängende Bearbeitung mehrerer Masken des Utility-Monitors erfordert.

independent DBH
independent DBH

siehe *Data Base Handler*

Index
index

baumartige Zugriffsstruktur mit Verweisen auf die *Sätze* einer *Tabelle*, die einer *Spalte* oder Spaltenkombination dieser Tabelle zugeordnet ist. Ein Index ordnet über eine *invertierte Liste* jedem Wert bzw. jeder Wertekombination der zugrundeliegenden Spalte(n) die Sätze zu, die diesen Wert bzw. diese Wertekombination in den betreffenden Spalten enthalten. Indizes dienen im Wesentlichen zur Beschleunigung der Datenwiedergewinnung.
Je nachdem, ob sich ein Index auf eine oder mehrere Spalten bezieht, spricht man von einem einfachen oder zusammengesetzten Index. Ein Index hat innerhalb des Schemas, in dem er definiert ist, einen eindeutigen Namen und liegt auf einem *Space*.
Ein Index wird mit der *SQL*-Anweisung CREATE INDEX erzeugt.
Für jede Spalte(nkombination), für die die Eindeutigkeitsbedingung definiert ist, benötigt SESAM/SQL einen Index. Wenn bereits ein Index mit CREATE INDEX für die betreffende(n) Spalte(n) erzeugt wurde, so wird dieser Index zusätzlich für die Eindeutigkeitsbedingung verwendet.
Ansonsten wird der benötigte Index implizit erzeugt. Der Name des implizit erzeugten Index beginnt mit UI, gefolgt von einer 16-stelligen Zahl.
Ein Index für die Spalten, die der *Primärschlüsselbedingung* genügen, wird auch Primärindex oder Primärschlüsselindex genannt, ein Index, der nicht Primärschlüsselindex ist, wird auch Sekundärindex genannt.

Index-Browsing (CALL-DML)
index browsing (CALL DML)

Funktion der *CALL-DML*, mit der die Häufigkeit von Attributwerten (siehe *Spalte*) festgestellt werden kann.

Index-Space
index space
Anwender-Space, in dem mindestens ein *Index* gespeichert ist.
Ein *Space* ist ein Index-Space einer bestimmten *Basistabelle*, wenn in ihm mindestens ein *Index* dieser *Basistabelle* gespeichert ist.

Indikatorvariable
indicator variable
spezielle *Benutzervariable* vom ganzzahligen numerischen *Datentyp* SMALLINT, die den Inhalt einer anderen Benutzervariablen beschreibt. Die Indikatorvariable zeigt an, ob die zugeordnete Benutzervariable den *NULL-Wert* enthält, oder ob bei der Übertragung von alphanumerischen Werten in die Benutzervariable Datenverlust aufgetreten ist.

INFORMATION_SCHEMA
INFORMATION_SCHEMA
siehe *Schema*

Informationsschema
information schema
siehe *Schema*

Inner Join
inner join
Join-Operation, bei der alle Sätze des *Kartesischen Produkts* in die *Ergebnistabelle* aufgenommen werden, die die angegebene Join-Bedingung erfüllen.

Integritätsbedingung
integrity constraint
eine Regel, die den durch den *Datentyp* festgelegten Wertebereich für eine *Spalte* oder mehrere Spalten einschränkt. Eine Integritätsbedingung schränkt somit die in einer Basistabelle zulässigen Sätze ein. Eine Integritätsbedingung kann entweder bei der Definition einer Spalte (*Spaltenbedingung*) oder als Eigenschaft einer *Tabelle* (*Tabellenbedingung*) vereinbart werden. SESAM/SQL kennt *CHECK-*, *Eindeutigkeits-*, *Primärschlüssel-*, *Referenz-* und *NOT NULL-Bedingungen*. Die Einhaltung einer Integritätsbedingung wird vom Datenbanksystem überwacht.

interner Wiederanlauf
internal restart
Wiederanlauf nach internen Fehlern geringeren Gewichts. Der *DBH* führt den internen Wiederanlauf durch, ohne dass der laufende Betrieb unterbrochen wird.

invertierte Liste
inverted list

einer *Spalte* einer *Tabelle* zugeordnete Verweisliste, in der zu jedem Spaltenwert angegeben ist, welche *Sätze* der Tabelle diesen Wert in der betreffenden Spalte enthalten. Entsprechend gibt es invertierte Listen für Spaltenkombinationen. Invertierte Listen werden benötigt im Zusammenhang mit Sekundärindizes (siehe *Index*).

Isolationslevel
isolation level

gibt an, wie stark das konsistente Lesen von Sätzen in einer *Transaktion* durch *konkurrierende Zugriffe* (ändernd) einer anderen Transaktion beeinträchtigt werden kann.
Es gibt folgende Isolationslevel: „READ UNCOMMITTED“, „READ COMMITTED“, „REPEATABLE READ“ und „SERIALIZABLE“. Je nach gewähltem Isolationslevel können die Phänomene „*dirty read*“, „*non-repeatable read*“ und „*phantoms*“ verhindert werden.

Join
join

Bei *SQL*: Der Join verknüpft zwei oder mehr Tabellen über die Werte der *Join-Spalte(n)*. In SESAM/SQL gibt es vier Formen des Joins: den *Cross Join*, den *Inner Join* und den *Outer Join* und den *Union Join*.
Bei *CALL-DML*: Der Join wird über die *Join-Suchfrage* realisiert.

Join-Attribut
join attribute

siehe *Join-Spalte*

Join-Ausdruck
join expression

Abfrage-Ausdruck, der das Schlüsselwort JOIN enthält und festlegt, welche *Tabellen* durch einen *Join* verknüpft werden sollen, welche Join-Operation gewünscht ist (*Cross*, *Inner*, *Outer* oder *Union Join*) und wie die Join-Bedingung lautet.

Join-Spalte
join column

Spalte einer *Tabelle*, deren Werte mit den Werten einer typverträglichen Spalte derselben oder einer anderen Tabelle paarweise verglichen werden. Ergibt der Vergleich den Wahrheitswert „wahr“, wird der aus dem *Join* resultierende Antwortsatz in die *Ergebnistabelle* aufgenommen.

Join-Suchfrage (CALL-DML)
search with join

verbindet zwei *logische Dateien*. Die Werte des Join-Attributs (vgl. *Join-Spalte*) der einen logischen Datei werden mit den Werten des Join-Attributs der anderen logischen Datei paarweise verglichen. Bei Gleichheit werden die entsprechenden Sätze aus den beiden logischen Dateien verknüpft und für das Ergebnis berücksichtigt.
Die beiden logischen Dateien können zur selben *Basistabelle* oder zu zwei verschiedenen Basistabellen gehören.

Julianische Tagesnummer
Scaliger's Julian day number

ganzzahliger Wert, der eine Anzahl von Tagen angibt, die seit einem bestimmten Anfangsdatum vergangen sind. Aus historischen Gründen ist das Anfangsdatum der 24. November des Jahres 4712 v. Chr. (gemäß dem Gregorianischen Kalender). Dieses Datum hat die Julianische Tagesnummer 0.
Die in SESAM/SQL erlaubten Daten von 0001-01-01 bis 9999-12-31 haben Julianische Tagesnummern von 1721426 bis 5373484.

Kartesisches Produkt
Cartesian product

Das Kartesische Produkt zweier *Tabellen* wird gebildet, indem jeder Satz der ersten Tabelle mit jedem Satz der zweiten Tabelle verkettet wird. Die Anzahl der Sätze des Kartesischen Produkts entspricht somit dem Produkt der Anzahl der Sätze der zugrundeliegenden Tabellen.
Die Bildung des Kartesischen Produkts für den Fall von mehr als zwei Tabellen erfolgt entsprechend.

Katalogkennung (CATID)
catalog ID (CATID)

bezeichnet das Pubset, auf dem die BS2000- bzw. SESAM/SQL-Dateien (*Spaces*, *datenbank-spezifische Dateien*, *DBH-spezifische Dateien*) gespeichert sind.
Die Katalogkennung wird dem Datenbank- bzw. Dateinamen in der Form :catid vorangestellt.

Kennwort (CALL-DML; SEPA)
password

von Bedeutung nur bei *Tabellen* der *Tabellenart* CALL-DML-Tabelle.
Dreistellige Zeichenfolge, die ein CALL-DML-Anwender angeben muss, wenn er auf geschützte CALL-DML-Tabellen zugreifen will.
Ein Kennwort kann den Zugriff einschränken auf einzelne Sätze einer CALL-DML-Tabelle, bestimmte *Attribute* oder eine bestimmte Zugriffsart (lesen, schreiben, lesen und schreiben). Das Kennwort und Angaben über die Zugriffsberechtigung stehen im *Kennwortkatalog*, der mit Hilfe des Dienstprogramms SEPA eingerichtet wird.

Kennwortkatalog (CALL-DML)
password catalog

Der Kennwortkatalog ist von Bedeutung bei *Tabellen* der *Tabellenart* CALL-DML-Tabelle. Er wird nur bei Zugriff über die *CALL-DML*-Schnittstelle ausgewertet.
Der Kennwortkatalog wird mit dem Dienstprogramm SEPA erstellt und gewartet und enthält eine Liste der Kennwörter sowie Angaben über Berechtigungen für den Zugriffsschutz.
Sofern für eine CALL-DML-Tabelle der SEPA-Zugriffschutz realisiert ist, ist dieser Tabelle ein eigener Kennwortkatalog zugeordnet, der auf demselben *Space* liegt.

Kommunikationsname
communication name

siehe *DBH-Name*

Konfiguration
configuration

zusammengehörige *DBHs*, Verteilkomponenten (siehe *SESAM/SQL-DCN*) und Anwenderprogramme auf einem Rechner, die unabhängig von anderen DBHs, Verteilkomponenten und Anwenderprogrammen auf diesem Rechner arbeiten.
Als Anwenderprogramm ist auch der *Utility-Monitor* einer Konfiguration zugeordnet.
Innerhalb des Rechners muss jede Konfiguration einen eindeutigen Konfigurationsnamen haben. Empfehlenswert ist die netzweite Eindeutigkeit.
In einem Rechner oder Rechnernetz können SESAM/SQL-Anwendungen gleichzeitig und unabhängig voneinander ablaufen, sofern sie jeweils Konfigurationen unterschiedlichen Namens angehören.

Konfigurationsdatei
configuration file
BS2000-Datei, die der Anwender erstellt, und in die er entweder Startparameter für den DBH oder Steueranweisungen für SESDCN oder Startparameter für das Konnektionsmodul eines Anwenderprogramms (DBCON) und/oder für den Utility-Monitor eintragen kann.
Eine Konfigurationsdatei für den DBH enthält ausschließlich DBH-Startanweisungen und -Optionen.
Eine Konfigurationsdatei für SESDCN enthält ausschließlich DCN-Steueranweisungen.
Eine Konfigurationsdatei für DBCON oder für den Utility-Monitor kann Startparameter für DBCON oder für den Utility-Monitor oder für beide Komponenten enthalten.

Konfigurationsname
configuration name
siehe *Konfiguration*

konkurrierende Transaktionen
concurrent transactions
siehe *konkurrierender Zugriff*

konkurrierender Zugriff
concurrent access
gleichzeitiger Zugriffsversuch von zwei oder mehreren Anwenderprogrammen auf einen *Satz*. Mit Hilfe von Transaktionen koordiniert SESAM/SQL den konkurrierenden Zugriff (konkurrierende Transaktionen).

Konnektionsmodul
connection module
Komponente, die die Verbindung zwischen *DBH* und Anwenderprogramm herstellt und in jedes Anwenderprogramm eingebunden werden muss.

Konsistenz
consistency
Widerspruchsfreiheit der gespeicherten Daten. Siehe auch *Consistency Check*.

Konsistenzlevel
consistency level
wurde durch den *Isolationslevel* abgelöst. Konsistenzlevel werden nur noch aus Gründen der Kompatibilität mit SESAM/SQL < V2.0 unterstützt.

Konsistenzpunkt
consistency point

Zeitpunkt, zu dem die Datenbank in einem logisch widerspruchsfreien (konsistenten) Zustand ist. Während einer DBH-Session mit *Transaktionssicherung* stellt der *DBH* zu definierten Zeitpunkten konsistente Datenbanken her. Diese Konsistenzpunkte sind bei der Transaktionssicherung für den Wiederanlauf als Rücksetzpunkte von Bedeutung.

Konstante
constant

siehe *Literal*

Korrelationsname
correlation name

Name, der innerhalb einer *SQL*-Anweisung für eine *Tabelle* zusätzlich vergeben oder für eine abgeleitete Tabelle neu vergeben werden kann.
Synonym: Synonym

Die Zeichenfolge `DO` wird in der FOR-Anweisung verwendet. Sie darf nicht als Korrelationsname angegeben werden. Sie können aber den Spezialnamen `"DO"` verwenden.

Ladeoption
load option

siehe *DBH-Option, DCN-Option*

linked-in DBH
linked-in DBH

siehe *Data Base Handler*

Literal
literal

Zeichenfolge, die einen festen Wert repräsentiert. SESAM/SQL unterscheidet alphanumerische und Spezial-Literale, numerische Literale und Zeit-Literale.
Synonym: Konstante

Little Endian

Eine Reihenfolge der Bytes einer Codierung im Speicher. Bei Little Endian liegt das niedrigstwertige Byte an der niedrigsten Speicheradresse.
Antonym: *Big Endian*.

Locksequenz
lock sequence

Locksequenzen spielen eine Rolle bei der Administration des DBH. Eine Locksequenz kann als eine Zeitspanne aufgefasst werden. Innerhalb dieser Spanne werden alle durch den Anwender angeforderten Sperren auf Datenbankverzeichnisse und Spaces aufrechterhalten. Die Locksequenz wird mit der DBH-Administrationsanweisung BEGIN-LOCK-SEQUENCE begonnen und mit der Anweisung END-LOCK-SEQUENCE wieder abgeschlossen. Diese abschließende Anweisung gibt die Sperren wieder frei.

logische Datei (CALL-DML)
logical file (CALL DML)

ein vom Anwender definierter Ausschnitt aus einer *Tabelle* der *Tabellenart* CALL-DML-Tabelle, der die verarbeitungsorientierte Sicht des Anwenders auf die CALL-DML-Tabelle darstellt.
Die logische Datei wird durch das *Dateikennzeichen* identifiziert.

logischer Datenbankname
logical database name

Name, mit dem ein Anwenderprogramm eine SESAM/SQL-Datenbank anspricht. Der logische Datenbankname kann der Name einer existierenden Datenbank sein. In diesem Fall ist der logische Datenbankname gleich dem *physikalischen Datenbanknamen*. Existiert zu dem logischen Datenbanknamen keine SESAM/SQL-Datenbank, erfolgt die Zuordnung zu einer existierenden Datenbank über den physikalischen Datenbanknamen im *SQL-Datenbankverzeichnis*.

logische Datensicherung
logical data saving

siehe *DA-Logging*

logisches After-Image
logical after image

Satz bzw. Satzteil innerhalb eines *Blocks* nach erfolgter Datenänderung. Als logische After-Images (LAIs) werden die Änderungen auf den *DA-LOG-Dateien* und *CAT-LOG-Dateien* protokolliert (im Rahmen des *Media-Recovery*).

logisches Before-Image
logical before image

Satz bzw. Satzteil innerhalb eines *Blocks* vor begonnener Datenänderung. Das logische Before-Image (LBI) enthält zusätzlich systeminterne Rücksetzinformation. LBIs dienen im Rahmen der *Transaktionssicherung* zum Rücksetzen offener *Transaktionen*.

lokale Fehler-Routine
local error handling routine

Mit der Definition lokaler Fehler-Routinen wird festgelegt, wie reagiert werden soll, wenn bei der Verarbeitung einer *Prozeduranweisung* im Rahmen der *COMPOUND-Anweisung* ein SQLSTATE ≠ '00000' gemeldet wird.

lokale Prozedurvariable
local procedure variable

Lokale Prozedurvariablen sind Variablen, die nur innerhalb der *COMPOUND-Anweisung* angesprochen werden können. Für sie wird ein Datentyp und ggf. ein Standardwert definiert. Sie besitzen keine Indikatorvariable.

lokaler Cursor
local cursor

Mit der Definition lokaler Cursor werden Cursor festgelegt, die nur innerhalb der *COMPOUND-Anweisung* angesprochen werden können.

Longlock
longlock

Zustand, in dem eine *Transaktion* für lange Zeit (Lock-Time) untätig ist und Betriebsmittel für andere Transaktionen sperrt. Der *DBH* löst Longlock-Situationen automatisch auf, indem er Transaktionen zurücksetzt, die für längere Zeit die Verarbeitung unterbrechen und damit den Lock-Time-Wert überschreiten. Der Lock-Time-Wert beträgt standardmäßig vier Minuten, kann aber modifiziert werden entweder über eine DBH-Option oder während des laufenden Betriebs über eine Administrationsanweisung.

Maske
form

ein auf dem Bildschirm dargestelltes Schema zur Anzeige und Eingabe von Daten. Masken werden für die Benutzeroberfläche des *Utility-Monitors*, des SESADMs und des SESMONs verwendet.

Maskenkurzbezeichnung
form short name

dreistelliger mnemotechnischer Name, der im Statusbereich der Masken des *Utility-Monitors* ausgegeben wird und der die aufgerufene Funktion wiederspiegelt.

Maskenname
form name

Name, der im Statusbereich der Masken des *Utility-Monitors* ausgegeben wird und der Auskunft über die aktuelle Funktion gibt.

Media-Recovery
media recovery

Maßnahme des *Recovery* zum Schutz vor Datenverlust, z.B. bei Hardwarefehlern. Das Media-Recovery umfasst sowohl die erforderlichen Maßnahmen zur Datensicherung (Erstellen von Sicherungsbeständen, *DA-Logging*), als auch die Maßnahmen zur Wiederherstellung des Datenbestandes (Rücksetzen, Reparatur).
Mit dem Rücksetzen erhält man einen intakten Datenbestand, wie er zum Zeitpunkt der Kopie-Erstellung vorlag. Die Reparatur, die alle seit der letzten Kopie-Erstellung in der *Datenbank* bzw. im *Space* angefallenen Änderungen berücksichtigt, erzeugt einen intakten Datenbestand, wie er unmittelbar vor der Ausfallsituation gegeben war.
Bei SESAM/SQL ist die kleinste Recovery-Einheit der *Space*.

Medienkatalog
media catalog

Der Medienkatalog enthält Speicherinformationen für bestimmte *DBH-spezifische Dateien* (Cursor-Datei, temporäre Arbeitsdateien, TA-LOG-Datei, WA-LOG-Datei).
Er beschreibt, auf welchem Datenträger, unter welcher *Katalogkennung* und mit welchen Speicherzuweisungen diese Dateien angelegt werden sollen. Der *Systemverwalter* kann den Medienkatalog über *DBH-Optionen* einrichten. Der Medienkatalog liegt nicht in Dateiform vor, sondern wird im *DBH* gehalten und ist nur für eine *DBH-Session* gültig.

Medientabelle
media table

In der Medientabelle sind die Dateieigenschaften und Speichergeräte für *datenbank-spezifische Dateien* hinterlegt. Die Medientabelle liegt auf dem *Catalog-Space*. Der Anwender kann den Inhalt der Medientabelle über die Views der Informationsschemata (siehe *Schema*) abfragen. Beim Anlegen einer datenbank-spezifischen Datei informiert sich der *DBH* in der Medientabelle, wo und wie die betreffende Datei angelegt werden soll. Die Medientabelle wird mit den *Utility-Anweisungen* CREATE/ALTER/DROP MEDIA DESCRIPTION FOR bearbeitet.

Mehrbenutzerbetrieb
multi-user operation

Betriebsart, bei der gleichzeitig mehrere Anwender mit einem *DBH* arbeiten.

Mehrdatenbankbetrieb
multi-db operation

Betriebsart, bei der jeder einzelne Anwender parallel maximal 254 Datenbanken bearbeiten kann.

Mengenfunktion
set function

Funktion, die einen Wert aus einer Menge von *Spaltenwerten* (AVG, MAX, MIN, SUM, COUNT) oder *Sätzen* (COUNT*) berechnet.

Metadaten
metadata

Daten, die ein Datenbanksystem zur Verwaltung der *Anwenderdaten* benötigt, z.B. Daten, die die Struktur der Anwenderdaten beschreiben. Die Metadaten sind im *Catalog-Space* der Datenbank abgelegt.

Migration
migration

überführt Datenbanken (Basistabellen), die mit SESAM/SQL-V1.x erstellt wurden, in *Basistabellen* einer SESAM/SQL-Datenbank der aktuellen Version. Die Migration erfolgt mit der *Utility-Anweisung* MIGRATE. Je nach Eigenschaft der zu migrierenden Datenbanken entstehen als Ergebnis der Migration *Tabellen* unterschiedlicher *Tabellenart*.

Mischbetrieb
mixed mode

Betriebsart einer Anwendung, die SESAM/SQL-Datenbanken sowohl über die *SQL*- als auch über die *CALL-DML*-Schnittstelle bearbeitet.

monadischer Operator
monadic operator

siehe *einstelliger Operator*

multiple Spalte
multiple column

Spalte, in der pro Datensatz mehrere Werte desselben *Datentyps* abgespeichert werden können. Einen solchen Wert nennt man Ausprägung einer multiplen Spalte.
Synonyme: multiples Attribut, multiples Feld

multiples Attribut
multiple attribute

siehe *multiple Spalte*

multiples Feld
multiple field

siehe *multiple Spalte*

Multitasking
multitasking

Mit der Multitasking-Architektur kann der *DBH* bei höheren Performance-Anforderungen auch mit mehreren Tasks geladen werden. Dadurch lässt sich bei Multiprozessor-Anlagen die DBH-Last auf mehrere Prozessoren verteilen.

Multithreading-Verfahren
multithreading

Verfahren, durch das der *DBH* die Zentraleinheit (CPU) so intensiv wie möglich nutzen kann.
Beim Multithreading-Verfahren bearbeitet der DBH parallel mehrere Aufträge unter Verwendung sogenannter Threads. In jedem Thread sind Informationen über den gegenwärtigen Zustand eines bestimmten Auftrags hinterlegt. Muss ein Auftrag auf den Abschluss eines Eingabe/Ausgabe-Vorgangs warten, nutzt der DBH die CPU für die Verarbeitung eines anderen Auftrags.
Die Anzahl Threads und somit die Anzahl paralleler Aufträge legt der *Systemverwalter* per *DBH-Option* fest.

nicht-signifikanter Attributwert
non-significant attribute value

bei *Tabellen* der *Tabellenart* CALL-DML-Tabelle ein *Attributwert*, der ausschließlich aus *Defaultwertzeichen* besteht und in der Datenbank nicht gespeichert wird. Das Defaultwertzeichen darf nicht verwechselt werden mit dem *SQL-Defaultwert*.

Noncharacter
noncharacter

Unicode-Zeichen, die permanent für interne Zwecke reserviert sind, und die in SQL-Daten nicht vorkommen dürfen. Es sind die 66 Unicode-Zeichen U&'\FDDx', U&'\FDEx', U&'\+0xFFFE', U&'\+0xFFFF', U&'\+10FFFE' und U&'\+10FFFF', wobei x eine Hexadezimalziffer ist. Ihre Verwendung in Literalen oder Hostvariablen führt in *SQL* zu einem Fehler.

non-repeatable read
non-repeatable read

Phänomen, das in Abhängigkeit vom eingestellten *Isolationslevel* auftreten kann:
Eine *Transaktion* liest einen *Satz*, ohne ihn gegen den Zugriff anderer Transaktionen zu sperren. Während diese Transaktion noch offen ist, ändert oder löscht eine zweite Transaktion diesen Satz und schreibt diese Änderung oder Löschung fest (COMMIT WORK, Ende Transaktion). Ein erneuter Zugriffsversuch der ersten Transaktion auf den Satz liefert dann veränderte Werte oder bleibt erfolglos.

Normalisieren
normalize

Umwandlung einer Unicode-Zeichenfolge, bei der Zeichen und Zeichenfolgen, die auf mehrere Arten in Unicode dargestellt werden können, in eine einheitliche Darstellung gebracht werden. In der Normalization Form D werden dabei Zeichen soweit wie möglich zerlegt, in der Normalization Form C werden auch zusammengesetzte Zeichen (z.B. „Ä“) verwendet. In *SQL* sollten Unicode-Zeichenfolgen in der Normalization Form C gespeichert werden.
Normalisieren wird von SESAM/SQL in der SQL-Funktion NORMALIZE() angeboten.

NOT NULL-Bedingung
NOT NULL constraint

Integritätsbedingung, die fordert, dass eine *Spalte* keinen *NULL-Wert* enthält. Ist eine NOT NULL-Bedingung für eine Spalte vereinbart, so verhindert SESAM/SQL, dass ein Satz der *Tabelle* in dieser Spalte den NULL-Wert enthält.

NULL-Wert
NULL value

spezieller Wert, der anzeigt, dass der Inhalt einer *Spalte* oder das Ergebnis eines *Ausdrucks* undefiniert oder unbekannt ist.

Outer Join
outer join

JOIN-Operation, bei der zusätzlich zu den Antwortsätzen eines *Inner Joins* auch je ein Satz in die *Ergebnistabelle* aufgenommen wird, für die in der *Join-Spalte* der einen Tabelle kein zur Join-Spalte der anderen Tabelle passender Wert gefunden wurde. Die fehlenden Werte werden in der Ergebnistabelle durch *NULL-Werte* dargestellt. Mit den Schlüsselwörtern LEFT, RIGHT und FULL wird festgelegt, aus welchen der beiden Tabellen die zusätzlichen Ergebnissätze stammen.

Partition
partition

Bereich (*Sätze*) einer *partitionierten Tabelle*, der auf einem Space gespeichert wird. Die Partitionsgrenzen werden über Intervalle des *Primärschlüssel*wertes bestimmt.

partitionierte Tabelle
partitioned table

Basistabelle mit einer besonderen physikalischen Struktur, den *Partitionen*. Maximal 16 Partitionen können für eine partitionierte Tabelle definiert werden.

PBI-Datei
PBI file

datenbank-spezifische Datei. PBI-Dateien benötigt SESAM/SQL, um bei der Online-Erstellung von SESAM-Sicherungsbeständen (als Plattenkopie oder mit ARCHIVE) mit der Utility-Anweisung COPY einen konsistenten Stand der Sicherungsbestände gewährleisten zu können. SESAM/SQL protokolliert die *physikalischen Before-Image*s (PBI) aller Blöcke, die während des Kopiervorgangs durch SQL- oder CALL-DML-Anweisungen geändert werden.

persistente Daten
persistent data

Daten werden als persistent bezeichnet, wenn sie so lange existieren, bis sie vom Anwender gelöscht werden. Im Gegensatz dazu stehen alle Daten, die bei *Transaktions*- oder *Session*-Ende oder bei Ende eines ESQL-COBOL-Programmlaufs automatisch gelöscht werden.

Phantoms
phantoms

Phänomen, das in Abhängigkeit vom eingestellten *Isolationslevel* auftreten kann:
Eine *Transaktion* wählt mit einer Abfrage (z.B. *SELECT-Ausdruck*, *Suchfrage*) Sätze aus einer Tabelle aufgrund einer bestimmten Bedingung aus. Während diese Transaktion noch offen ist, nimmt eine zweite Transaktion Sätze in die Tabelle auf, die ebenfalls dieser Bedingung genügen. Wiederholt die erste Transaktion dieselbe Abfrage, so enthält das Ergebnis auch die neu aufgenommenen Sätze.

physikalisches After-Image
physical after image

Block nach erfolgter Datenänderung. Das physikalische After-Image (PAI) wird benötigt zur Herstellung der physikalischen Konsistenz beim Wiederanlauf.

physikalisches Before-Image
physical before image

Block vor begonnener Datenänderung. Mit Hilfe von physikalischen Before-Images (PBIs) wird bei der Online-Erstellung von Sicherungsbeständen (*Utility-Anweisung* COPY) ein konsistenter Zustand der *Datenbank* gewährleistet. Außerdem wird das PBI zur Herstellung der physikalischen Konsistenz beim Wiederanlauf benötigt.

physikalischer Datenbankname
physical database name

Name einer existierenden SESAM/SQL-Datenbank. Der physikalische Datenbankname kann verschieden sein vom *logischen Datenbanknamen*, mit dem ein Anwenderprogramm eine SESAM/SQL-Datenbank anspricht.

Planpuffer
plan buffer

Hauptspeicherbereich, den der *DBH* für das Zwischenspeichern von *SQL-Zugriffsplänen* reserviert und verwaltet.

Pragma
pragma

Spezieller SQL-Kommentar. Gibt Hinweise für die Ausführung einer SQL- oder *Utility-Anweisung*. Ein Pragma wirkt sich auf die gesamte Anweisung aus, einschliesslich der verwendeten Views. Das Pragma PREFETCH wirkt sich sogar auf alle Operationen mit einem *Cursor* aus.

Prädikat
predicate

in einer *Suchbedingung* enthaltene Operation, die den Wahrheitswert wahr, falsch oder unbekannt liefert.

Prefetch-Cursor
prefetch cursor

Cursor, für den über das *Pragma* PREFETCH der *Schubmodus* eingeschaltet wurde.

Prefetch-Puffer
prefetch buffer

Speicherbereich, in dem die von einer FETCH NEXT-Anweisung im *Schubmodus* eingelesenen Sätze zwischengepuffert werden.

Primärindex
primary index

siehe *Index*
Synonym: Primärschlüsselindex

Primärschlüssel
primary key

eine oder mehrere *Spalten*, die einen *Satz* in einer *Tabelle* eindeutig identifizieren. Ein Primärschlüssel wird durch eine *Primärschlüsselbedingung* festgelegt. Pro Tabelle darf maximal ein Primärschlüssel definiert sein.

Primärschlüsselbedingung
primary key constraint

Integritätsbedingung für eine *Spalte* bzw. Spaltenkombination einer *Tabelle*. Die Primärschlüsselbedingung fordert, dass die *Eindeutigkeitsbedingung* erfüllt ist und außerdem die betreffende Spalte(nkombination) keinen *NULL-Wert* enthält. Pro Tabelle darf höchstens eine Primärschlüsselbedingung definiert sein.

Primärschlüsselindex
primary key index

siehe *Index*

Privilegien
privileges

legen fest, welche *SQL*-Anweisungen ein Benutzer auf einer *Tabelle* oder *Spalte* ausführen darf. SESAM/SQL unterscheidet zwischen SELECT-, INSERT-, UPDATE-, DELETE-, EXECUTE- und REFERENCES-Privilegien. Neben diesen Privilegien kennt SESAM/SQL auch noch eine Reihe von *Sonder-Privilegien*.
Synonym: Zugriffsrecht

Prozedur
procedure

In einer Prozedur werden Abläufe von *SQL*-Anweisungen in der Datenbank gespeichert, die später mit einem einzigen Aufruf ausgeführt werden können. Eine Prozedur ist vergleichbar mit einem Unterprogramm, das vollständig, also ohne Datenaustausch mit dem Anwendungsprogramm, im *DBH* abläuft. Prozeduren werden mit der *SQL*-Anweisung CALL aufgerufen. Sie haben Eingabe- und Ausgabeparameter.
Synonym: Stored Procedure

Prozeduranweisung
procedure statement

Eine Prozeduranweisung ist eine *SQL*-Anweisung, die innerhalb einer *Prozedur* verwendet werden darf. Jede *Prozedur* enthält genau eine Prozeduranweisung. SESAM/SQL kennt folgende Prozeduranweisungen:

– *SQL*-Anweisungen ohne *Cursor*:
  SELECT, INSERT, UPDATE, DELETE, MERGE, CALL
– *SQL*-Anweisungen mit *Cursor*:
  OPEN, FETCH, UPDATE, DELETE, CLOSE
– *SQL*-Anweisungen zur Prozedurkontrolle:
  COMPOUND, IF, LOOP, LEAVE, SET

Prozedurparameter
procedure parameter

Aufrufparameter einer *Prozedur*. Für jeden Prozedurparameter muss ein Name, ein Datentyp und die Art des Prozedurparameters (IN, OUT oder INOUT) angegeben werden.

Projektion
projection

Auswahl von *Spalten* aus einer oder mehreren *Tabellen* zur Übernahme in eine *Ergebnistabelle*.

Puffer
buffer

Hauptspeicherbereich zur Zwischenspeicherung (Pufferung) von *Blöcken*, Sicherungsinformationen und anweisungsbezogenen Daten. Ziel der Pufferung ist es, Platteneingabe und -ausgabe zu minimieren.
Die Pufferverwaltung des *DBH* regelt die Verdrängung von *Blöcken* bei Pufferüberlauf.

Pufferverwaltung
buffer management

siehe *Puffer*

Pubset
pubset

Pubsets sind Sätze gemeinschaftlicher Platten und ein Ablageort für Dateien in BS2000. Ein herausragendes Merkmal des Pubsets ist, dass neben den Dateien selbst auch alle für die Dateiverwaltung erforderlichen Metadaten (Dateikatalog, Benutzerkatalog, usw.) in ihm enthalten sind.

qualifizierter Name
qualified name

dient der Identifizierung eines *SQL*-Objekts durch Angabe eines übergeordneten SQL-Objekts. Qualifizierte Namen werden gebildet, indem der Name des übergeordneten SQL-Objekts durch einen Punkt getrennt vorangestellt wird. So können beispielsweise gleichnamige *Tabellen*, die verschiedenen *Schemata* zugeordnet sind, durch Qualifizierung mit dem Schema-Namen unterschieden werden.

Recovery
recovery

Oberbegriff für alle Sicherungs- und Wiederherstellungsverfahren bei *Datenbanksystemen*. Recovery umfasst einerseits Maßnahmen für einen schnellen Wiederanlauf des Datenbanksystems bei festgestellten Inkonsistenzen, zum anderen bieten Recovery-Maßnahmen Schutz vor Datenverlust bei Systemausfall und bei Hardwarefehlern.
Im Einzelnen werden unterschieden:
- *Transaktionsrecovery* (transaction recovery)
- *Wiederanlauf* (extern/intern) des DB- bzw. DB/DC-Systems (system recovery)
- *Media-Recovery* (media recovery)

REF-Wert
ref value

Jedes *BLOB-Objekt* besitzt einen eindeutigen REF-Wert, der das Objekt referenziert. SESAM/SQL benutzt diesen REF-Wert, um auf das BLOB-Objekt zuzugreifen.

Referenzbedingung
referential constraint

Integritätsbedingung, die durch den *Fremdschlüssel* definiert ist.
Die Referenzbedingung fordert bei einspaltigen Fremdschlüsseln, dass jeder vom *NULL-Wert* verschiedene Wert des Fremdschlüssels einer *Tabelle* (sog. referenzierende Tabelle) als Wert einer bestimmten Spalte in einer anderen Tabelle (sog. referenzierte Tabelle) vorkommt, die der *Eindeutigkeitsbedingung* genügt.
Besteht der Fremdschlüssel aus mehreren Spalten, so muss jede darin vorkommende Wertekombination, die keinen NULL-Wert enthält, in einer bestimmten Spaltenkombination der referenzierten Tabelle auftreten. Diese Spaltenkombination muss der Eindeutigkeitsbedingung genügen.

Regulärer Ausdruck
regular expression

Reguläre Ausdrücke sind genau definierte Suchmuster. Sie sind ein mächtiges Mittel, um große Datenbestände nach komplexen *Suchausdrücken* zu durchsuchen. Sie werden seit langem z.B. in der Programmiersprache Perl eingesetzt. Sie können in SESAM/SQL im *Prädikat* LIKE_REGEX verwendet werden.

Relation
relation siehe *Tabelle*

remote Zugriff
remote access

bei der *verteilten Verarbeitung* mit SESAM/SQL-DCN: Zugriff eines Anwenderprogramms auf eine Datenbank, die einer anderen *Konfiguration* angehört.

Reorganisation
reorganization

Bei der Reorganisation werden logisch zusammengehörige *Blöcke* physikalisch zusammenhängend abgespeichert. Dabei wird die gewünschte *Freiplatzreservierung* berücksichtigt. Reorganisation erfolgt auch bei physikalischer Verlagerung des *Spaces* nach Zuweisung einer anderen *Storage Group* durch den *Datenbankverwalter*. Reorganisation und gegebenenfalls physikalische Verlagerung erfolgen mit der *Utility-Anweisung* REORG.

Replikat
replication

Kopie einer Datenbank, die im DML-Betrieb ausschließlich für Wiedergewinnung verwendet werden kann. Es handelt sich um den definierten Stand einer Originaldatenbank, der mit der DA-LOG-Datei aktualisiert und wieder in einen definierten Zustand gebracht werden kann. Ein Replikat kann außerdem zum Reparieren oder Rücksetzen einer Originaldatenbank oder zum Erzeugen eines neuen Originals verwendet werden.

Routine
routine

Oberbegriff für *Prozedur* und *User Defined Function (UDF)* in SESAM/SQL. Prozeduren und UDFs haben einen fast identischen Funktionsumfang, sie unterscheiden sich aber durch die Art ihres Aufrufes und ihrer Rückkehr-Information.

Sammelkonsistenzpunkt
group commit

siehe *Group Commit*

Satz
row (SQL)

geordnete Folge von Werten, die einer *Zeile* einer *Tabelle* entspricht.
Synonym: *Tupel*, *Zeile*

Satzauswahl
selection

siehe *Selektion*

Satzsperre
record lock

sperren eines Datenbanksatzes durch eine *Transaktion* gegen den Zugriff anderer Transaktionen (siehe *Transaktionsparallelität*).

Schema
schema

Schemata sind im *Catalog-Space* der *Datenbank* enthalten.
Man unterscheidet die anwenderdefinierten Schemata und die Informationsschemata.
Anwenderdefinierte Schemata enthalten Metadaten, die den formalen Aufbau der *Basistabellen* und *Views* einer Datenbank beschreiben. Außerdem enthalten die anwenderdefinierten Schemata weitere Metadaten, wie z.B. Angaben zu *Privilegien*.
Ein anwenderdefiniertes Schema hat einen Namen und einen Eigentümer, der durch einen *Berechtigungsschlüssel* für dieses Schema ausgewiesen ist. Es wird mit der *SQL*-Anweisung CREATE SCHEMA erstellt und kann durch weitere Anweisungen (z.B. CREATE TABLE) modifiziert werden.
Bei den Informationsschemata unterscheidet man INFORMATION_SCHEMA und SYS_INFO_SCHEMA, die für jede Datenbank vorhanden sind. Sie ermöglichen dem Anwender Zugriff auf die in den anwenderdefinierten Schemata abgelegten *Metadaten*, wie z.B. Beschreibungen von *Basistabellen*, *Views*, *Integritätsbedingungen*, *Privilegien*, usw.
Das INFORMATION_SCHEMA kann jeder Anwender mit SQL-Mitteln abfragen. Das SYS_INFO_SCHEMA enthält systemspezifische Daten und ist nur dem *universellen Benutzer* zugänglich.

Schlüssel
key

Eine Folge von Bits, die in Algorithmen zum *Verschlüsseln* oder *Entschlüsseln* verwendet wird. Das Verschlüsselungsverfahren *AES*, das in SESAM/SQL verwendet wird, benutzt denselben Schlüssel für beide Operationen. Der Schlüssel muss daher geheim gehalten werden.
In SESAM/SQL ist der Schlüssel eine alphanumerische Zeichenkette mit 16 Byte (128 bit).
Es gibt auch Verschlüsselungsverfahren mit unterschiedlichen Schlüsseln zum Verschlüsseln und Entschlüsseln, wobei nur einer der beiden Schlüssel geheim gehalten werden muss (public key cryptography).

Schubmodus
block mode

Modus, der über das *Pragma* PREFETCH für einen Cursor eingeschaltet werden kann. Bei eingeschaltetem Schubmodus veranlasst die erste Anweisung FETCH NEXT, dass der DBH mehrere Sätze der Cursortabelle in einen Zwischenpuffer (*Prefetch-Puffer*) einliest. Der Anwender erhält mit der ersten FETCH-Anweisung nur den ersten übertragenen Satz; weitere Ausführungen dieser FETCH-Anweisung übertragen jeweils den nächsten Satz aus dem Zwischenpuffer, bis der Zwischenpuffer leer ist und eine weitere FETCH-Anweisung wieder zur Übertragung mehrerer Sätze führt. Durch den Schubmodus kann die Bearbeitung eines Cursors wesentlich beschleunigt werden.

Sekundärindex
secondary index

siehe *Index*

SELECT-Ausdruck
query specification

durch das Schlüsselwort SELECT eingeleiteter *Abfrageausdruck* mit eingeschränkter Syntax.

SELECT-Liste
select list

legt fest, welche Spalten die *Ergebnistabelle* enthalten soll. Sie enthält Namen von *Spalten* aus einer oder mehreren *Tabellen* sowie beliebige andere *Ausdrücke*.

Selektion
selection

Auswahl von *Sätzen* aus einer oder mehreren *Tabellen* nach einer vorgegebenen Bedingung.

Server
server

siehe *Client-Server-Architektur*

Service-Task
service task

Eine von ggf. mehreren BS2000-Tasks, auf die SESAM/SQL CPU-intensive Aktionen auslagert. Service-Tasks stehen z.B. für Funktionen der Datenbankverwaltung und für das Sortieren von Zwischenergebnissen zur Verfügung.

SESAM-CLI (Call Level Interface)
SESAM CLI
: Prozedurale Schnittstelle für komplexe Operationen von SESAM/SQL, die sich nur umständlich oder gar nicht mit *SQL*-Anweisungen ausführen lassen.

SESAM-Sicherungsbestand
SESAM backup copy
: Kopie, die mit der SESAM-Anweisung COPY erzeugt worden ist.

SESAM/SQL-DBH
SESAM/SQL DBH
: siehe *Data Base Handler*

SESAM/SQL-DCN
SESAM/SQL DCN
: Zusatzprodukt, das den SESAM/SQL-Betrieb mit *verteilten Datenbanken* ermöglicht.

SESAM/SQL-spezifische Anweisungen
SESAM/SQL-specific statements
: Anweisungen in *SQL*-Syntax, die nicht in der *SQL-Norm* enthalten sind, z.B. *Utility-Anweisungen*.

SESDCN-Steueranweisung
SESDCN control statement
: Die SESDCN-Steueranweisungen umfassen die *DCN-Optionen* sowie die Anweisungen zur Definition der *Verteilregel*.
Der *Systemverwalter* gibt die SESDCN-Steueranweisungen beim Starten der *Verteilkomponente SESDCN* an.

Session
session
: siehe *DBH-Session, SQL-Session*

Single-System-Image
single system image
: Die Taskfamilie des SESAM/SQL-DBH stellt sich dem *Systemverwalter* als eine Einheit dar. Die einzelnen Tasks sind für die Administration nicht relevant (siehe *Multitasking*).

Sonder-Privilegien
special privileges

legen die Datendefinitions- und Administrationsanweisungen fest, die ein *SQL*-Benutzer ausführen darf. SESAM/SQL unterscheidet die Sonder-Privilegien CREATE USER, CREATE SCHEMA, CREATE STOGROUP, UTILITY und USAGE ON STOGROUP.

Space
space

benannter Speicherbereich, der als BS2000-Datei realisiert und einer *Storage Group* zugeordnet ist. Pro *Datenbank* gibt es einen *Catalog-Space* und ein oder mehrere *Anwender-Spaces*.
Spaces sind die Organisationseinheiten für *Reorganisation* und *Recovery*.

Space-Liste
space list

Sicherungseinheit im Rahmen des *Media-Recovery*, bestehend aus mehreren *Anwender-Spaces*. Der Anwender kann für eine *Recovery* selber eine Space-Liste zusammenstellen aus mehreren Anwender-Spaces mit gemeinsamem Zeitstempel. Spaces mit gemeinsamem Zeitstempel entstehen bei einer gemeinsamen Copy-Anweisung. Der Zeitstempel gibt den Zeitpunkt der Kopieerstellung an und muss für alle in der Space-Liste zusammengefassten *Anwender-Spaces* gleich sein.

Space-Set
space set

Sicherungseinheit im Rahmen des *Media-Recovery*, bestehend aus mehreren *Anwender-Spaces*. Ein Space-Set entsteht durch eine gemeinsame COPY-Anweisung mehrerer Anwender-Spaces und wird durch einen Zeitstempel identifiziert. Der Zeitstempel gibt den Zeitpunkt der Kopieerstellung an und ist für alle im Space-Set zusammengefassten *Anwender-Spaces* gleich.

Spalte
column

Bestandteil einer *Tabelle*. Jede Spalte besitzt einen Namen und einen *Datentyp* und enthält *Spaltenwerte* dieses Datentyps. Man unterscheidet *einfache* und *multiple Spalten*.
Im Handbuch „CALL-DML Anwendungen“ wird statt Spalte der Begriff Attribut verwendet.
Synonym: Attribut

Spaltenbedingung
column constraint

Integritätsbedingung, die bei der Tabellendefinition als Eigenschaft einer *Spalte* vereinbart wird. Die Spaltenbedingung wird bei der Definition der betreffenden Spalte angegeben.

Spaltenwert
column value

Unter einem Spaltenwert versteht man bei einer einfachen *Spalte* einen atomaren Wert vom Datentyp der Spalte. Bei einem multiplen Wert ist ein Spaltenwert ein *Aggregat* mit Werten des Datentyps der Spalte.
Im Handbuch „CALL-DML Anwendungen“ wird statt Spaltenwert der Begriff Attributwert verwendet.
Synonym: Attribut

Sperreinheit
locking granularity

siehe *Sperrgranulat*

Sperrgranulat
locking granularity

Einheit für Datenobjekte, z.B. Satz oder Tabellen, auf die Sperren gesetzt werden können.

Sperrkonzept
locking concept

siehe *Transaktionsparallelität*

Spiegelplatte
mirror disk

Plattensatz, der aus mindestens zwei Platten mit identischem Inhalt besteht.

SQL
SQL (Structured Query Language)
: ist die am weitesten verbreitete Sprache zur Bearbeitung von relationalen Datenbanken. Im Gegensatz zu den prozeduralen Sprachen nicht-relationaler Datenbanksysteme ist SQL deskriptiv, d.h., der Anwender beschreibt in einer mengenorientierten Form das Ergebnis einer Datenbankoperation und nicht die zu diesem Ergebnis führenden Schritte. SQL bietet umfangreiche Sprachmittel zur Datendefinition, Datenmanipulation, Transaktionsverwaltung, Verwaltung von Zugriffsrechten usw. sowie als Embedded SQL (ESQL) die Möglichkeit, innerhalb einer Wirtssprache (z.B. COBOL) mit *eingebetteten SQL*-Anweisungen auf eine Datenbank zuzugreifen.
SQL wurde erstmals 1987 von der International Organization for Standardization normiert. SESAM/SQL basiert auf dem aktuellen Standard, kurz *SQL-Norm* genannt. SESAM/SQL enthält neben dem Sprachkern von SQL („Core SQL“) auch optionale Sprachelemente von SQL und ergänzende Funktionen, die in SQL fehlen (*Utility-Anweisungen*).

SQL-Norm
SQL standard
: Kurzbezeichnung in den Handbüchern zu SESAM/SQL für den aktuellen internationalen SQL-Standard, derzeit ISO/IEC 9075:2003 („SQL03“).

SQLCODE
SQLCODE
: siehe *SQL-Statuscode*

SQLSTATE
SQLSTATE
: siehe *SQL-Statuscode*

SQL-Datenbankverzeichnis
SQL database catalog list
: enthält einen Eintrag für jede *Datenbank*, die während einer *DBH-Session* bearbeitet werden soll. Insbesondere wird dem *DBH* über das SQL-Datenbankverzeichnis mitgeteilt, welche Datenbank welcher BS2000-Benutzerkennung zugeordnet ist.
Der *Systemverwalter* richtet das SQL-Datenbankverzeichnis per *DBH-Startanweisung* ein. Einträge in das SQL-Datenbankverzeichnis kann der *Systemverwalter* über entsprechende *Administrationsanweisungen* hinzufügen oder löschen.

SQL-Defaultwert
SQL default value

Wert, der in der DEFAULT-Klausel der CREATE TABLE-Anweisung als Defaultwert für eine *Spalte* angegeben wird und dann als Vorbelegung für die Werte dieser Spalte dient. Der SQL-Defaultwert darf nicht verwechselt werden mit dem *nicht-signifikanten Attributwert*.

SQL-Modus
SQL mode

Modus, in dem sich ein Anwenderprogramm befindet, sobald es eine *SQL*-Anweisung in einer Transaktion ausführt, und solange die Transaktion nicht beendet ist.

SQL-Returncode
SQL return code

siehe *SQL-Statuscode*

SQL-Scan
SQL scan

Teil eines *SQL-Zugriffsplans*, der sich auf genau eine Basistabelle bezieht. Einen SQL-Zugriffsplan erzeugt der SESAM/SQL-DBH aus einer *SQL*-Anweisung. Synonym: Scan

SQL-Session
SQL session

Folge von *SQL*-Anweisungen, die ein Benutzer vom Zeitpunkt des Verbindungsaufbaus mit SESAM/SQL bis zur Abmeldung ausführt.

SQL-Statuscode
SQL status code

Der SQL-Statuscode (SQLSTATE) enthält Informationen darüber, ob eine *SQL*-Anweisung ohne Fehler ausgeführt wurde oder ob die Bearbeitung zu einem Fehler führte.
Eine fehlerfreie Ausführung wird durch die Statuscodes für „erfolgreiche Ausführung“, „Warnung“ oder „keine Daten vorhanden“ angezeigt.
Nach einer fehlerhaften Bearbeitung der *SQL*-Anweisung enthält der SQL-Statuscode Informationen über die Art des Fehlers.
Die SQL-Returncodes (SQLCODEs), die nicht mehr in Standard SQL enthalten sind und nur aus Gründen der Kompatibilität zu früheren SESAM/SQL-Versionen noch unterstützt werden, haben die gleiche Funktion wie die SQL-Statuscodes, sind aber weniger aussagekräftig.

SQL-Tabelle
SQL table

siehe *Tabellenart*

SQL-Zugriffsplan
SQL access plan

Auswertungsvorschrift, die der DBH intern aus jeder *SQL*-Anweisung erstellt. Ein *SQL*-Zugriffsplan besteht aus mindestens einem, meist aber mehreren Teilbereichen, den sogenannten *SQL-Scans*. Das optimierte Format eines Scan bildet schließlich die *Arbeitsleiste*.

Storage Group
storage group

benannte Menge von Platten derselben *Katalogkennung*. Alle Platten müssen denselben Gerätetyp haben. Über Storage Groups kann der Anwender steuern, wo die *Spaces* (BS2000-Dateien) angelegt werden. Jeder Space ist einer Storage Group zugeordnet, auf deren Platten der Space angelegt wird.

Storage Structure Language (SSL)
storage structure language (SSL)

Anweisungen zur Verwaltung der Speicherstruktur; mit ihrer Hilfe können *Storage Groups*, *Spaces* und *Indizes* erzeugt bzw. bearbeitet werden.
Storage Structure Language ist nicht Bestandteil der *SQL-Norm*.

Suborders
suborders

Bei der Bearbeitung von *SQL*-Anweisungen sind Suborders gleichbedeutend mit *SQL-Scans*, also Teilbereiche eines *SQL-Zugriffsplans*. Im CALL-DML-Fall sind Suborders *logische Dateien* für CALL-DML-Aufträge.

Suchbedingung
search condition

SQL-Sprachmittel, das einen Wahrheitswert wahr, falsch oder unbekannt liefert. Es besteht aus *Prädikaten*, die mit logischen Operatoren AND, OR und NOT verbunden werden können. Die Suchbedingung dient dazu, die Menge der Sätze der Ergebnistabelle einzuschränken (z.B. bei Verwendung in der WHERE-Klausel eines *Abfrage-Ausdrucks*).
Bei Verwendung in einer *CHECK-Bedingung* schränkt die Suchbedingung die zulässigen Werte für die betreffende(n) *Spalte(n)* ein.

Suchfrage (CALL-DML)
search (CALL DML)

zentrale *CALL-DML*-Anweisung für die *Datenwiedergewinnung.*

Surrogate Pairs
surrogate pairs
: In *UTF-16* eine Folge von zwei Code Units, die ein Unicode-Zeichen mit Code Point größer als U&'\FFFF' darstellen.
NX'D800' bis NX'DBFF': Leading Surrogate;
NX'DC00' bis NX'DFFF': Trailing Surrogate.
Ein Paar Leading und Trailing Surrogate bilden die Code Points von U&'\+010000' bis U&'\+10FFFF' ab.

symbolischer Attributname (CALL-DML)
symbolic attribute name (CALL DML)
: dreistelliger Bezeichner für ein Attribut, der bei *CALL-DML*-Anweisungen das Attribut identifiziert.

Synonym
synonym
: siehe *Korrelationsname*

SYS_INFO_SCHEMA
SYS_INFO_SCHEMA
: siehe *Schema*

Systemadministration
system administration
: siehe *Systemverwaltung*

Systemadministrator
system administrator
: siehe *Systemverwalter*

System-Benutzerkennung
system user identification
: identifiziert im BS2000 einen TIAM- oder openUTM-Anwender.
Ein TIAM-Anwender weist sich aus durch den (symbolischen) Rechnernamen und die BS2000-Benutzerkennung.
Ein UTM-Anwender weist sich aus durch den (symbolischen) Rechnernamen, den Namen der UTM-Anwendung und den KDCSIGN- bzw. LSES-Namen.
Um mit einer SESAM/SQL-Datenbank arbeiten zu können, muss dem TIAM- bzw. UTM-Anwender ein *Berechtigungsschlüssel* zugeordnet sein (siehe *Systemzugang*).

System-Data-Buffer
system data buffer
: Hauptspeicherbereich, den der *DBH* für *Systemdaten* reserviert und verwaltet.

Systemverwalter
system administrator
für die *Systemverwaltung* zuständige Person oder Personengruppe.
Synonym: SESAM/SQL-Systemverwalter, Systemadministrator

Systemverwaltung
system administration
Aufgabengebiet, das das Starten und Beenden des *DBHs* sowie die Betreuung der laufenden *DBH-Session* umfasst.

Systemzugang
system entry
Paar, bestehend aus *System-Benutzerkennung* und *Berechtigungsschlüssel (eines SQL-Benutzers)*. Der Systemzugang berechtigt zum Zugriff auf eine SESAM/SQL-Datenbank. Mit Ausnahme des *universellen Benutzers* wird ein Systemzugang mit der *SQL*-Anweisung CREATE SYSTEM_USER eingerichtet.

TA-LOG-Dateien
TA LOG files
Sicherungsdateien, in denen Informationen für die *Transaktionssicherung*, das *Transaktionsrecovery* sowie den externen und internen *Wiederanlauf* gespeichert sind. Unter anderem sind in den TA-LOG-Dateien *Before-Images* und *After-Images* gespeichert.
Synonym: Transaktionssicherungsdateien

TA-LOG-Puffer
TA LOG buffer
Hauptspeicherbereich, in dem der *DBH* Sicherungsinformationen für die Transaktionssicherung zwischenspeichert. Bei Pufferüberlauf bzw. beim Schreiben eines Konsistenzpunkts überträgt der DBH die Daten des TA-LOG-Puffers auf die *TA-LOG-Dateien*.

Tabelle
table
Eine Tabelle ist eine zweidimensionale Anordnung von Datenelementen, die aus *Zeilen* (horizontal) und *Spalten* (vertikal) besteht.
Man unterscheidet *Basistabellen*, *Views*, *Ergebnistabellen* und *abstrakte Tabellen*.
Synonym: Relation

Tabellenart
table style

gibt an, ob eine *Basistabelle* über die *SQL*-Schnittstelle und/oder die *CALL-DML-Schnittstelle* bearbeitet werden kann. Die Tabellenart ist insbesondere von Bedeutung im Zusammenhang mit der *Migration*. Folgende Tabellenarten werden unterschieden:

– Tabellen, die nur mit *CALL-DML* bearbeitet werden können (Nur-CALL-DML-Tabellen)
– Tabellen, die mit CALL-DML und (eingeschränkt) mit *SQL* bearbeitet werden können (CALL-DML/SQL-Tabellen)
– Tabellen, die nur mit *SQL* bearbeitet werden können (SQL-Tabellen)
– Tabellen, die nur zur Speicherung von *BLOB-Objekt*en dienen (*BLOB-Tabellen*).

Nur-CALL-DML-Tabellen und CALL-DML/SQL-Tabellen werden unter der Tabellenart CALL-DML-Tabelle zusammengefasst. Bei CALL-DML-Tabellen muss für jedes *Attribut* ein *nicht-signifikanter Attributwert* vereinbart sein.

Tabellenbedingung
table constraint

Integritätsbedingung, die mit CREATE TABLE oder ALTER TABLE als Eigenschaft einer *Basistabelle* definiert wird. Sie kann eine *Eindeutigkeitsbedingung*, eine *Referenzbedingung* oder eine *CHECK-Bedingung* sein.

Table-Space
table space

Anwender-Space, in dem mindestens eine *Basistabelle* gespeichert ist. Der Table-Space einer bestimmten *Basistabelle* ist derjenige *Space*, in dem diese *Basistabelle* gespeichert ist.

Teilhaberbetrieb
transaction mode

Im Teilhaberbetrieb greifen mehrere Anwender gleichzeitig mit demselben Anwenderprogramm auf eine oder mehrere Datenbanken zu. Die Koordination der Teilhaber erfolgt unter Steuerung des Transaktionsmonitors openUTM oder des Datenkommunikationssystems DCAM.

Teilnehmerbetrieb
time-sharing mode

Im Teilnehmerbetrieb arbeiten mehrere Anwender unabhängig voneinander jeweils mit einem eigenen Anwenderprogramm im *Batch-* oder *Dialogbetrieb*.

Teilreplikat
partial replication

Replikat, das nicht sämtliche *Spaces* der Originaldatenbank enthält.

Thread
thread

siehe *Multithreading-Verfahren*

Transaktion
transaction

Folge von zusammengehörigen Anweisungen, die eine *Datenbank* von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt. Eine Transaktion wird entweder vollständig oder überhaupt nicht ausgeführt.
Für die Transaktionsverarbeitung gibt es spezielle Anweisungen:
CALL-DML kennt jeweils eine Anweisung zum Öffnen, Schließen (Festschreiben) und Rücksetzen einer CALL-DML-Transaktion.
In *SQL* gibt es spezielle Anweisungen nur zum Beenden und Rücksetzen einer SQL-Transaktion; in openUTM-Anwendungen müssen für diesen Zweck die entsprechenden UTM-Aufrufe verwendet werden. Der Beginn einer SQL-Transaktion wird nicht mit einer speziellen *SQL*-Anweisung festgelegt: die erste transaktionseinleitende Anweisung nach dem Rücksetzen oder Festschreiben der vorherigen Transaktion oder dem Programmstart wertet SESAM/SQL als Transaktionsbeginn.

transaktionseinleitende SQL-Anweisung
transaction-initiating SQL statement

Transaktionseinleitende *SQL*-Anweisungen sind alle Anweisungen außer den Anweisungen DECLARE CURSOR, SET TRANSACTION, SET CATALOG, SET SCHEMA, SET SESSION AUTHORIZATION, PERMIT, WHENEVER, ALTER TABLE mit Pragma UTILITY MODE=ON sowie *Utility-Anweisungen*.

Transaktionsrecovery
transaction recovery

Maßnahme des *Recovery* zum Rücksetzen einer einzelnen *Transaktion* nach Anwenderfehlern.

Transaktionsmodus
access mode of transaction

wird mit der *SQL*-Anweisung SET TRANSACTION vereinbart und legt fest, ob innerhalb einer *Transaktion* Sätze nur gelesen (READ ONLY) oder auch geändert werden dürfen (READ WRITE).

Transaktionsorientierte Verarbeitung
transaction processing

gewährleistet, dass Zugriffe auf eine Datenbank nur innerhalb von *Transaktionen* möglich sind. Damit ist die Konsistenz einer *Datenbank* zu jedem Zeitpunkt sichergestellt.

Transaktionsparallelität
transaction concurrency
gleichzeitiger Zugriff mehrerer *Transaktionen* auf denselben Datenbestand. Durch das Sperrkonzept von SESAM/SQL ist auch bei parallelem Zugriff die *Konsistenz* der bearbeiteten Daten gewährleistet. Im Standardfall sperrt jede Transaktion die Sätze, auf die sie zugreift, gegen den ändernden Zugriff anderer Transaktionen. Die Sperre wird erst bei Transaktionsende aufgehoben. Der Anwender kann jedoch das Sperrverhalten bedarfsgerecht modifizieren und so den Grad der Transaktionsparallelität und damit den Durchsatz erhöhen. Hierzu dient in SQL-Anwendungen der *Isolationslevel*.

Transaktionssicherung
transaction management
Die Transaktionssicherung stellt Mechanismen zur automatischen Behandlung von Fehlersituationen zur Verfügung und stellt im Fehlerfall die *Konsistenz* der Datenbestände über Rücksetzmechanismen wieder her. Somit bildet die Transaktionssicherung die Grundlage für die *Recovery*-Maßnahmen *Transaktionsrecovery* und *Wiederanlauf* (extern/intern).

Transaktionssicherungsdatei
transaction log file
siehe *TA-LOG-Datei*

Transaktionsstatus
transaction status
siehe *Transaktionsmodus*

Transfer-Container
transfer container
Hauptspeicherbereich, den der DBH für *SQL-Scans* und für Frage- und Antwortbereiche von *logischen Dateien* reserviert und verwaltet.

Transcodierung
transcoding
Umwandlung einer National-Zeichenkette vom Zeichensatz UTFE in den Zeichensatz UTF-16.
In *SQL* wird ein Transcoding explizit mit der SQL-Funktion TRANSLATE() ausgeführt. Siehe auch: *Transliteration*.

Transliteration
transliteration

Umwandlung einer alphanumerischen Zeichenkette in eine National-Zeichenkette und umgekehrt.
In *SQL* wird eine Transliteration explizit mit der SQL-Funktion TRANSLATE() ausgeführt. Implizite Transliterationen werden, falls erforderlich, bei der *SQL*-Anweisung ALTER TABLE ALTER COLUMN sowie bei den *Utility-Anweisungen* LOAD und UNLOAD ausgeführt.
Siehe auch: *Transcodierung*.

Tuning
tuning

Optimieren der Performance (Antwortzeitverhalten, Betriebsmittelverbrauch, Durchsatz) des *Datenbanksystems*.

Tupel
tuple

siehe *Satz*

Two-Phase-Commit
two-phase commit

beendet eine *Transaktion* bei der *verteilten Verarbeitung* in zwei Phasen: zunächst leitet der Auftrag „vorläufiges Transaktionsende“ (PREPARE TO COMMIT) den Abschluss der Transaktion ein. Alle an der Transaktion beteiligten Komponenten versuchen nun, einen Zustand zu erreichen, in dem die Transaktion sowohl zurückgesetzt als auch beendet werden kann. Wenn mindestens ein an der Verteilung beteiligter *DBH* diesen Zustand nicht erreicht, wird anschließend die gesamte Transaktion zurückgesetzt; sonst wird in einer zweiten Phase der endgültige Auftrag „Ende Transaktion“ erteilt.

Union Join
union join

Join-Operation, deren *Ergebnistabelle* sowohl die Sätze der Tabelle links als auch die Sätze der Tabelle rechts vom UNION-Operator enthält, jeweils ergänzt um die auf *NULL-Werte* gesetzten Spalten der anderen Tabelle.

UNIQUE-Index
UNIQUE index

siehe *Index*

Universal Character Set der Länge 2 (UCS-2)

Bei *UTF-16* werden alle Unicode-Zeichen zwischen NX'0000' bis NX'FFFF' durch 2 Bytes codiert. Die Zeichen in diesem Bereich werden auch als Universal Character Set der Länge 2 (UCS-2) bezeichnet.

universeller Benutzer
universal user

SQL-Benutzer mit umfassenden *Privilegien*, der bei der Datenbankgenerierung mit der *Utility-Anweisung* CREATE CATALOG für jede *Datenbank* festgelegt wird. Er wird durch den ersten *Systemzugang* beschrieben.
Mit der *SQL*-Anweisung CREATE SYSTEM USER kann der universelle Benutzer weitere Systemzugänge mit den Privilegien eines universellen Benutzers schaffen, indem er anderen *System-Benutzerkennungen* seinen Berechtigungsschlüssel zuordnet.

unkorrelierter Funktionsaufruf
uncorrelated function call

Funktionsaufruf einer *UDF* mit konstanten Eingabewerten. Konstante Eingabewerte beziehen sich nicht auf die SQL-Anweisung, die den Funktionsaufruf enthält,

Unterabfrage
subquery

in Klammern eingeschlossener *Abfrage-Ausdruck*, der entweder einen einfachen Spaltenwert, einen Datensatz bestehend aus mehreren Spaltenwerten, eine *Spalte* oder eine *Tabelle* als Ergebnis liefert.

User Defined Function (UDF)
User Defined Function (UDF)

In einer UDF werden Abläufe von *SQL*-Anweisungen in der Datenbank gespeichert, die später mit einem einzigen Aufruf ausgeführt werden können. Eine UDF ist vergleichbar mit einem Unterprogramm, das vollständig, also ohne Datenaustausch mit dem Anwendungsprogramm, im *DBH* abläuft.
UDFs können in fast allen Ausdrücken durch ihren Funktionsaufruf verwendet werden. Sie haben Eingabeparameter und liefern einen Rückgabewert.

User-Data-Buffer
user data buffer

Hauptspeicherbereich, den der *DBH* für *Anwenderdaten* reserviert und verwaltet.

UTF-8

Unicode-Codierung, definiert vom Unicode-Konsortium.
UTF-8 verwendet eine variable Anzahl von Bytes (ein bis vier Bytes) zur Codierung der Unicode-Zeichen. Die Byte-Darstellung der ASCII-Zeichen bleibt unverändert. Bei einem Zeichen, das in mehreren Bytes codiert ist, repräsentiert keines der Einzelbytes ein gültiges Zeichen.

UTF-16

Unicode-Codierung, definiert vom Unicode-Konsortium.
Bei UTF-16 sind alle Unicode-Zeichen zwischen NX'0000' bis NX'FFFF' durch 2 Bytes codiert. Alle Zeichen oberhalb von NX'FFFF' werden durch 4 Byte dargestellt, so genannte *Surrogate Pairs*.
SESAM/SQL speichert Werte in Spalten vom Datentyp N[VAR]CHAR intern im UTF-16 Format und verwendet UTF16 auch für National- Werte in Hostvariablen.

UTF-EBCDIC (UTFE)

Unicode-Codierung für Maschinen, die den EBCDIC-Zeichensatz verwenden. Dabei werden die Unicode Zeichen von NX'0000' bis NX'009F' durch das entsprechende EBCDIC-Zeichen dargestellt, und alle anderen Unicode-Zeichen durch Folgen mit 2 bis 5 Bytes.
In SESAM/SQL kann bei den *Utility-Funktionen* UNLOAD und LOAD mit DELIMITER_FORMAT Unicode mit UTF-EBCDIC-Codierung verwendet werden.

Utility-Anweisung
utility statement

Utility-Anweisungen sind Anweisungen in *SQL*-Syntax, die Dienstprogrammfunktionen für die *Datenbankverwaltung* bereitstellen. So gibt es u.a. Utility-Anweisungen zum Generieren, Laden, Entladen, Kopieren, Reorganisieren und Migrieren von Datenbanken.
Utility-Anweisungen leiten keine Transaktion ein, sie sind nicht Bestandteil der *SQL-Norm*.

Utility Mode
utility mode

Modus, den der Anwender beim Hinzufügen, Ändern oder Löschen einer oder mehrerer *Spalten* einer *Basistabelle* über das *Pragma* UTILITY MODE einstellen kann. Der Utility Mode bewirkt, dass die zugehörige *SQL*-Anweisung keine Transaktion einleitet und das *Transaktionssicherung* ausgeschaltet wird.

Utility-Monitor
utility monitor

Tool für die *Datenbankverwaltung* mit folgenden Funktionen:
- Ausführen von *SQL*- und *Utility-Anweisungen*.
- Auswerten von *INFORMATION_SCHEMA* und *SYS_INFO_SCHEMA* des Catalog-Space.
- Lesen und ändern der *Metadaten* von *Catalog-Recovery-Datei* und *Spaces*.

Außerdem ermöglicht der Utility-Monitor den Aufruf des Administrationsprogramms SESADM als Unterprogramm.
Der Utility-Monitor wird wahlweise im Dialog über eine Menüoberfläche bedient oder liest seine Eingaben aus einer Anweisungsdatei.

Vektor
vector

Datenobjekt, das sich aus einer bestimmten Anzahl von Werten gleichen *Datentyps* zusammensetzt.

Vektorvariable
vector variable

Benutzervariable für einen *Vektor*, die zur Aufnahme der Ausprägungen einer *multiplen Spalte* dient.

Verschlüsselung
encryption

Die Umwandlung von Klartext in Geheimtext mit Hilfe eines Verschlüsselungs-Algorithmus und einem *Schlüssel*.
Wird in SESAM/SQL mit der SQL-Funktion ENCRYPT() ausgeführt.

Verteilkomponente SESDCN
SESDCN distribution component

zentraler Bestandteil des Produkts *SESAM/SQL-DCN.* Wesentliche Aufgaben der Verteilkomponente SESDCN sind Aufbau und Verwaltung der *Verteilregel* sowie Entgegennahme und Weiterleitung von *Remote-Zugriffen*. Daneben erfüllt SESDCN administrative und überwachende Funktionen.

verteilte Datenbanken
distributed databases

Datenhaltung, bei der die zugehörigen Datenbanken auf verschiedenen Rechnern liegen (verteilte Datenhaltung) in Verbindung mit *verteilter Verarbeitung*.
Verteilte Datenbanken ermöglichen z.B. Anwendungen, bei denen eine Datenbank jeweils auf dem Rechner liegt, auf dem die Daten am häufigsten benötigt werden. Diese Daten können von Anwenderprogrammen anderer Rechner ebenfalls transaktionsgesichert abgefragt oder geändert werden.

verteilte Datenhaltung
distributed data management
siehe *verteilte Datenbanken*

verteilte Transaktion
distributed transaction
siehe *globale Transaktion*

verteilte Verarbeitung
distributed processing
wird bei SESAM/SQL durch das Zusatzprodukt *SESAM/SQL-DCN* ermöglicht. Bei der verteilten Verarbeitung kann ein Anwenderprogramm innerhalb einer Session mit mehr als einem *DBH* transaktionsgesichert zusammenarbeiten. Die verteilte Verarbeitung kann innerhalb einer Konfiguration, konfigurationsübergreifend innerhalb eines Rechners oder rechnerübergreifend erfolgen. Dabei bleibt dem Anwenderprogramm verborgen, mit welchem DBH es gerade arbeitet. Die Zuordnung eines DBHs zu einer Datenbank sowie die Verteilung der Datenbanken im Rechnernetz erfolgt über die *Verteilregeln*.
Die verteilte Verarbeitung ist insbesondere von Bedeutung, wenn die bearbeiteten Datenbanken auf verschiedenen Rechnern liegen (*verteilte Datenbanken*). Unabhängig von der Verteilung von Datenbanken mit *SESAM/SQL-DCN* kann eine openUTM-Anwendung mit UTM-D auf mehrere Rechner verteilt werden.

Verteilregel
distribution rule
Für die *verteilte Verarbeitung* mit *SESAM/SQL-DCN* muss die Verteilung der Datenbanken auf die einzelnen Rechner bekannt sein. Außerdem muss innerhalb eines Rechners jede Datenbank einer *Konfiguration* zugeordnet sein.
In den Verteilregeln gibt der *Systemverwalter* jeweils an, auf welchem Rechner eine bestimmte *Datenbank* liegt, über welchen *DBH* der Zugriff auf diese Datenbank erfolgen soll und welche *Verteilkomponente* Zugriffe von Anwenderprogrammen anderer Konfigurationen auf die Datenbank weiterleitet.

View
view
aus einer oder mehreren *Tabellen* abgeleitete benannte Tabelle. Ein View wird mit der *SQL*-Anweisung CREATE VIEW durch einen *Abfrage-Ausdruck* definiert. Die durch den Abfrage-Ausdruck spezifizierte *Ergebnistabelle* wird jedesmal neu ermittelt, wenn der View in einer SQL-Anweisung angesprochen wird.
Somit enthält die Ergebnistabelle stets die aktuellen Werte der Datenbank.
Synonym: Benutzersicht

Vorgang
conversation

Ein Vorgang ist eine zusammengehörige Folge von Transaktionen. Für die Dauer des Vorgangs sind Betriebsmittel der *SQL-Session* wie z.B. gespeicherte Cursor ansprechbar. Diese Betriebsmittel des Vorgangs unterliegen keinerlei Sicherungsmaßnahmen, d.h., sie verschwinden mit dem Ende oder Abbruch der *DBH-Session*.

– Ein DB-Vorgang enthält 0 bis n DB-Transaktionen.
– Ein UTM-Vorgang enthält 1 bis n UTM-Transaktionen.
– Ein DB-Vorgang kann genau einem openUTM-Vorgang zugeordnet sein, oder er kann sich über mehrere UTM-Vorgänge erstrecken.
– Ein SQL-Vorgang entspricht einem DB-Vorgang, falls dies ein reiner SQL-Vorgang ist, oder einem DB-Teilvorgang, wenn er Bestandteil eines gemischten Vorgangs ist.

WA-LOG-Datei
WA LOG file

In dieser Datei sind Informationen zur Steuerung und Absicherung der DBH-Session und des DBH-Wiederanlaufs gespeichert. Jedem SESAM/SQL-DBH ist eine WA-LOG-Datei eindeutig zugeordnet. Die WA-LOG-Datei enthält u.a.:

– die DBH-Optionen, die der DBH für den Wiederanlauf übernimmt
– die physikalischen Before-Images
– Informationen, die den Fortschritt des Wiederanlaufs beschreiben
– Information zur Synchronisation mit openUTM.

Wiederanlauf
restart

Recovery-Maßnahme zur Wiederaufnahme des Betriebs nach Fehlersituationen.
Transaktionen, die zum Zeitpunkt des Fehlers offen sind, setzt der *DBH* auf den letzten, gültigen *Konsistenzpunkt* zurück, so dass alle beteiligten Datenbanken in einem konsistenten Zustand vorliegen. Je nachdem, ob die laufende *DBH-Session* unterbrochen wird oder nicht, führt der DBH den *externen* oder *internen* Wiederanlauf durch. Voraussetzung für den Wiederanlauf ist, dass der DBH mit *Transaktionssicherung* läuft.

Wiedergewinnung
retrieval

siehe *Datenwiedergewinnung*

Work-Container
work container

Hauptspeicherbereich, den der *DBH* für das Zwischenspeichern von *Arbeitsleisten* reserviert und verwaltet.

Zeile
row
siehe *Satz*

Zugriffsrechte
access authorization
siehe *Privilegien*

zusammengesetzter Index
compound index
siehe *Index*
Synonym: Compound Index

zusammengesetzter Primärschlüssel
compound primary key
Primärschlüssel, der aus einer Kombination mehrerer *Spalten* besteht.
Synonym: Compound Key

zweistelliger Operator
dyadic operator
Operator mit zwei Operanden, beispielsweise die Operatoren der Grundrechenarten +, -, * und /.
Synonym: dyadischer Operator

Literatur

Die Handbücher finden Sie im Internet unter *http://manuals.ts.fujitsu.com*. Handbücher, die mit einer Bestellnummer angezeigt werden, können Sie in auch gedruckter Form bestellen.

SESAM/SQL-Server (BS2000)
SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen
Benutzerhandbuch

SESAM/SQL-Server (BS2000)
SQL-Sprachbeschreibung Teil 2: Utilities
Benutzerhandbuch

SESAM/SQL-Server (BS2000)
CALL-DML Anwendungen
Benutzerhandbuch

SESAM/SQL-Server (BS2000)
Datenbankbetrieb
Benutzerhandbuch

SESAM/SQL-Server (BS2000)
Utility-Monitor
Benutzerhandbuch

SESAM/SQL-Server (BS2000)
Meldungen
Benutzerhandbuch

SESAM/SQL-Server (BS2000)
Performance
Benutzerhandbuch

WebTA-Zugang für SESAM/SQL
(Produktdokument, auch auf dem Handbuch-Server verfügbar)

ESQL-COBOL (BS2000)
ESQL-COBOL für SESAM/SQL-Server
Benutzerhandbuch

SESAM-DBAccess
Server-Installation, Administration (nur auf dem Handbuch-Server verfügbar)

SESAM-KLDS (BS2000)
Benutzerhandbuch

DRIVE/WINDOWS (BS2000)
Programmiersystem
Benutzerhandbuch

DRIVE/WINDOWS (BS2000)
Programmiersprache
Sprachbeschreibung

DRIVE/WINDOWS (BS2000)
Lexikon der DRIVE-SQL-Anweisungen für SESAM/SQL
Referenzhandbuch

DRIVE/WINDOWS (UNIX-Systeme)
Lexikon der DRIVE-Anweisungen
Referenzhandbuch

DRIVE (BS2000)
Ergänzungsband
Benutzerhandbuch

openUTM
Konzepte und Funktionen
Benutzerhandbuch

openUTM (BS2000, UNIX-Systeme, Windows NT)
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

openUTM (BS2000)
Einsatz von openUTM-Anwendungen unter BS2000
Benutzerhandbuch

openUTM (BS2000)
Anwendungen generieren und betreiben
Benutzerhandbuch

ARCHIVE (BS2000)
Benutzerhandbuch

EDT (BS2000)
Anweisungen
Benutzerhandbuch

FHS (BS2000)
Benutzerhandbuch

HSMS (BS2000)
Hierarchisches Speicher Management System
Benutzerhandbuch

SECOS (BS2000)
Security Control System - Zugangs- und Zugriffskontrolle
Benutzerhandbuch

SECOS (BS2000)
Security Control System - Beweissicherung
Benutzerhandbuch

SDF (BS2000)
Dialogschnittstelle SDF
Benutzerhandbuch

SHC-OSD (BS2000)
Storage Management für BS2000
Benutzerhandbuch

SNMP Management
SNMP Management für BS2000
Benutzerhandbuch

SPACEOPT (BS2000)
Optimierung und Reorganisation von Platten
Benutzerhandbuch

Unicode in BS2000
Übersichtshandbuch

XHCS (BS2000)
8-bit-Code- und Unicode-Unterstützung im BS2000
Benutzerhandbuch

Stichwörter

Im Stichwortverzeichnis verweisen **halbfette** Seitenzahlen auf die Hauptfundstellen von Stichwörtern und *kursive* Seitenzahlen auf Beispiele. Es gilt folgende Sortierreihenfolge: Symbole vor Ziffern vor Buchstaben. Satzzeichen sind Symbole.

D

E

S

V

W