Deutsch

UDS/SQL V2.9B

Aufbauen und Umstrukturieren

Benutzerhandbuch

Juni 2019

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 bs2000services@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2015

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

Copyright und Handelsmarken

Inhaltsverzeichnis

Aufbauen und Umstrukturieren

1 Einleitung

Das **U**niverselle **D**atenbank-**S**ystem UDS/SQL ist ein Datenbanksystem für hohe Durchsatzanforderungen. Es basiert auf dem Strukturkonzept von CODASYL, geht aber in seinen Möglichkeiten weit darüber hinaus und bietet koexistent auf dem gleichen Datenbestand das Relationenmodell an.

Zur Auswertung und Änderung der Daten stehen COBOL-DML, CALL-DML und SQL (ISO-konform) zur Verfügung. COBOL-DML-Anweisungen sind in die COBOL-Sprache integriert, die CALL-DML kann aus jeder Programmiersprache aufgerufen werden, SQL-Anweisungen können innerhalb von DRIVE-Programmen angewendet oder über eine ODBC-Schnittstelle genutzt werden.

UDS/SQL verhindert durch wirksame, flexibel einsetzbare Schutzmechanismen unberechtigte Zugriffe auf die Datenbank und garantiert Vertraulichkeit, Integrität und Verfügbarkeit. Diese Mechanismen sind mit dem Transaktionsmonitor openUTM abgestimmt.

Das Datensicherungskonzept von UDS/SQL schützt die Datenbestände wirkungsvoll vor Zerstörung und Verlust. Dabei werden UDS/SQL- eigene Mechanismen wie Logging veränderter Information mit BS2000-Funktionen wie DRV (Dual Recording by Volume) kombiniert.

Unter Einsatz des Zusatzproduktes UDS-D können Datenbestände in BS2000-Rechnernetzen verarbeitet werden. UDS/SQL garantiert dabei die netzweite Konsistenz der Daten. In Verbindung mit openUTM-D bzw. openUTM (Unix /Linux/Windows) lässt sich verteilte Transaktionsverarbeitung sowohl in BS2000-Rechnernetzen als auch im Verbund von BS2000 und anderen Betriebssystemen realisieren. UDS/SQL kann als Datenbank in Client-Server-Lösungen über SQL-Gateway bzw. über ODBC-Server eingesetzt werden.

UDS/SQL bietet durch seine Architekturmerkmale (z. B. Multitasking, Multithreading, DB-Cache) und durch seine vielseitigen Strukturierungsmöglichkeiten einen sehr hohen Durchsatz.

1.1 Konzept der UDS/SQL-Dokumentation

Dem Abschnitt „Wegweiser durch die Handbuchreihe“ entnehmen Sie, welche Handbücher und welche Teile daraus Ihrem Informationsbedürfnis entsprechen. Ein Fachwortverzeichnis liefert Kurzdefinitionen der im Text benutzten Fachwörter.
Außer über das Inhaltsverzeichnis können Sie die Antworten auf Ihre Fragen gezielt über das Stichwortverzeichnis und über Kolumnentitel nachschlagen.

Wegweiser durch die Handbuchreihe

Das Datenbanksystem UDS/SQL ist im Wesentlichen in fünf Handbüchern dokumentiert:

- UDS/SQL Entwerfen und Definieren
- UDS/SQL Anwendungen programmieren
- UDS/SQL Aufbauen und Umstrukturieren
- UDS/SQL Datenbankbetrieb
- UDS/SQL Sichern, Informieren und Reorganisieren

Weitere Handbücher zu UDS/SQL und Zusatzprodukten finden Sie auf "Weitere Handbücher".

Als Einstieg dient Ihnen das Handbuch „Entwerfen und Definieren", Kapitel 2 und 3; hier werden erläutert:

- die Gründe für den Einsatz von Datenbanken
- das Datenbankmodell der CODASYL
- das Relationenmodell unter Berücksichtigung von SQL
- eine Abgrenzung der Modelle
- die Koexistenz der verschiedenen Datenbankmodelle bei einer UDS/SQL-Datenbank
- die charakteristischen Eigenschaften von UDS/SQL

Der weitere Umgang mit den Handbüchern richtet sich nach Ihren Vorkenntnissen und Aufgaben. Die Tabelle 1 hilft Ihnen dabei, den richtigen Weg durch die Handbücher zu finden.

Beispiele

Angenommen, Ihre Aufgabe ist es, in COBOL-DML zu programmieren, so finden Sie in der zweiten Zeile der Tabelle 1 unter „Aufgaben des Anwenders“ die Spalte „COBOL/CALL-DML Programm“. Im Handbuch „Entwerfen und Definieren" brauchen Sie dann für Ihre Arbeit folgende Kapitel:

Allgemeines	E = zum Einstieg
Schema-DDL	D = zur Detailinformation
SSL	D = zur Detailinformation
Subschema-DDL	L = zum Lernen der Funktionen

Welche Kapitel Sie aus den weiteren Handbüchern brauchen, erfahren Sie in der gleichen Spalte.

Wenn Sie dagegen als Datenbankadministrator für den Datenbankbetrieb zuständig sind, orientieren Sie sich bitte in der Spalte „Verwalten und Bedienen“.

Inhalt der fünf Haupthandbücher	Aufgaben des Anwenders							
	Entwerfen und Definieren	COBOL/ CALL-DML Programm.	SQL-Program-mieren	Aufbauen und Umstrukt.	Verwalten und Bedienen	Arbeiten mit openUTM	Arbeiten mit IQS	Arbeiten mit UDS-D
Einleitung	E	–	–	–	–	E	E	–
Allgemeines	E	E	E	E	E	E	–	–
Entwurf der Datenbank	E	–	–	–	–	–	–	–
Schema-DDL	L	D	–	L	L	–	–	–
SSL	L	D	–	L	L	–	–	–
Subschema DDL	L	L	–	L	L	–	–	–
Relationales Schema	L	–	D	–	–	–	–	–
Aufbau der Seiten	D	–	–	D	D	–	–	–
Aufbau der Sätze und Tabellen	D	–	–	D	D	–	–	–
Nachschlageteil	S	–	–	S	–	–	–	–
Handbuch UDS/SQL Anwendungen programmieren								
Einleitung	–	E	–	–	–	E	E	–
Einführung	–	E	–	–	–	–	–	–
Transaktionskonzept	–	L	–	L	L	D	D	–
Currency-Tabelle	–	L	–	L	L	–	–	–
Funktionen der DML	D	L	–	L	–	–	–	–
Anwenden der DML	–	L	–	D	–	–	–	–
Nachschlageteil COBOL-DML	–	L	–	–	–	–	–	–
Nachschlageteil CALL-DML	–	L	–	–	–	–	–	–
Testen von DML-Funktionen mit DMLTEST	–	L	–	–	–	–	–	–
Handbuch UDS/SQL Aufbauen und Umstrukturieren								
Einleitung	–	–	–	E	–	E	E	–
Überblick	–	–	–	E	E	–	–	–
Datenbank aufbauen	–	–	–	L	–	–	–	–
Zugriffsberechtigungen festlegen	–	–	–	L	–	–	–	–
Daten speichern und entladen	D	–	–	L	–	D	–	–
Datenbank umstrukturieren	D	–	–	L	–	–	–	–
Datenbankobjekte umbenennen	D	–	–	L	–	–	–	–
Datenbank umstellen	D	–	–	L	–	–	–	–
Handbuch UDS/SQL Datenbankbetrieb								
Einleitung	–	–	–	–	E	E	E	–

Der Database Handler	–	–	–	–	L	–	–	D
Ladeparameter des DBH	–	–	–	–	L	–	–	D
Administration	–	–	–	–	L	–	–	D
Hochverfügbarkeit	–	–	–	–	E	–	–	–
Ressourcen-Erweiterung und Umorganisation im laufenden Betrieb	D	–	–	–	E	–	–	–
Datenbank sichern und wiederherstellen im Fehlerfall	D	–	–	D	L	D	–	D
Leistungsoptimierung	–	–	–	–	D	–	–	D
Nutzung der BS2000-Funktionalität	–	–	–	–	D	–	–	–
Der SQL-Vorgang	–	–	–	–	L	–	–	–
UDSMON	–	–	–	–	D	–	–	–
Einsatz von IQS	–	–	–	L	D	–	D	–
Einsatz von UDS-D	D	D	–	D	D	D	–	D
Funktionscodes der DML-Anweisungen	–	D	–	–	D	–	–	–
Handbuch UDS/SQL Sichern, Informieren und Reorganisieren								
Einleitung	–	–	–	–	E	E	E	–
Datenbank aktualisieren und rekonstruieren	D	–	–	D	L	D	–	–
Konsistenz einer Datenbank prüfen	–	–	–	–	L	–	–	–
Datenbankinformationen ausgeben	D	–	–	D	L	–	–	–
Online-Dienste durchführen	D	–	–	D	L	–	–	–
Datenbank reorganisieren	D	–	–	D	L	–	–	–
Wiederverwendung von freigewordenen Database Keys steuern	D	–	–	D	L	–	–	–
Weitere Handbücher								
UDS/SQL Meldungen	D	D	D	D	D	D	D	D
UDS/SQL Taschenbuch	S	S	–	S	S	S	S	S
IQS	–	–	–	D	D	–	L	–
ADILOS	–	–	–	–	D	–	L	–
KDBS	–	L [1]	–	D	–	–	–	–
SQL für UDS/SQL Sprachbeschreibung	–	–	D	–	D	–	–	–

Tabelle 1: Wegweiser durch die Handbucher

[1] only for COBOL-DML

E dient als Einstieg, wenn Sie bisher noch nichts mit UDS/SQL zu tun hatten

L in diesen Teilen der Handbücher steht das Lernen der Funktionen im Vordergrund

D hier können Sie hineinschauen, wenn Sie Detailinformationen suchen

S dient zum Nachschlagen von Syntaxregeln bei der praktischen Arbeit

Was Sie noch über die Handbücher wissen sollten

Literaturverweise finden Sie in Kurzform im Text. Finden Sie im Text z.B. (siehe Handbuch „Anwendungen programmieren“, CONNECT), so müssen Sie unter dem Stichwort CONNECT im Handbuch „Anwendungen programmieren“ nachschauen.
Der vollständige Handbuchtitel steht im Literaturverzeichnis.

UDS/SQL Meldungen

Das Handbuch enthält alle Meldungen, die UDS/SQL ausgibt. Die Meldungen sind aufsteigend nach Nummern oder bei einigen Dienstprogrammen alphabetisch sortiert.

UDS/SQL Taschenbuch

Das UDS/SQL-Taschenbuch enthält alle Übersichten zu den UDS/SQL-Funktionen und Formaten.

SQL für UDS/SQL Sprachbeschreibung

Das Handbuch beschreibt den SQL-DML-Sprachumfang von UDS/SQL.
Neben UDS/SQL-spezifischen Erweiterungen umfasst der beschriebene Sprachumfang die dynamische SQL als wesentliche Erweiterung der SQL-Norm.

1.2 Zielsetzung und Zielgruppen des Handbuchs

Das Handbuch ist für den Datenbankadministrator bestimmt, der die Aufgabe hat, die Datenbank aufzubauen, den Datenbankbetrieb zu planen sowie die Datenbank durch Umstrukturieren an die sich wandelnden Bedürfnisse des Betriebs anzupassen.

Der Datenbankadministrator sollte Kenntnisse haben über alle Schritte des Datenbankentwurfs (Datenbankdesign, Schema-, Subschema- und SSL-Erstellen) und des Erstellens der DB-Anwenderprogramme.

Außerdem muss er über gute BS2000-Kenntnisse verfügen.

1.3 Konzept des Handbuchs

Was enthält dieses Handbuch?

Einführend gibt dieses Handbuch eine Übersicht über die vom Datenbanksystem
UDS/SQL im laufenden Betrieb benötigten Dateien sowie die UDS/SQL-Dienstprogramme, die zum Aufbauen einer UDS/SQL-Datenbank nötig sind.
Anschließend finden Sie alle Schritte beschrieben, die erforderlich sind

- zum Umstrukturieren einer Datenbank,
- zum Umstellen einer Datenbank auf ein größeres Seitenformat.

Wie finden Sie sich im Handbuch zurecht?

Dem Abschnitt „Wegweiser durch die Handbuchreihe“ ("Konzept der UDS/SQL-Dokumentation") entnehmen Sie, welche Handbücher und welche Teile daraus Ihrem Informationsbedürfnis entsprechen. Ein Fachwortverzeichnis liefert Kurzdefinitionen der im Text benutzten Fachwörter.
Außer über das Inhaltsverzeichnis können Sie die Antworten auf Ihre Fragen gezielt über das Stichwortverzeichnis und über Kolumnentitel nachschlagen.

Readme-Datei

Funktionelle Änderungen der aktuellen Produktversion und Nachträge zu diesem Handbuch entnehmen Sie bitte ggf. der produktspezifischen Readme-Datei.

Readme-Dateien stehen Ihnen online bei dem jeweiligen Produkt zusätzlich zu den Produkthandbüchern unter https://bs2manuals.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 https://bs2manuals.ts.fujitsu.com.

1.4 Änderungen gegenüber den Vorgänger-Handbüchern

In der folgenden Tabelle 2 sind die wichtigsten Änderungen der Version UDS/SQL V2.9B gegenüber der Version V2.9 aufgeführt. Außerdem wird 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:

ENT	Entwerfen und Definieren	DBB	Datenbankbetrieb
ANW	Anwendungen programmieren	SIR	Sichern, Informieren und Reorganisieren
AUF	Aufbauen und Umstrukturieren	MEL	Meldungen

Thema	Handbuch	Kapitel
Änderungen in V2.9A		
FIND-/FETCH-7 mit DESCENDING KEY: Aufhebung der Einschränkung		
Die Einschränkung für DESCENDING KEY entfällt	ANW	7
Satzreferenzen in COBOL-Programmen		
Die neue DDL-Anweisung GENERATE-REC-REF generiert ein Datenfeld und Bedingungsnamen für den Zugriff auf Satzreferenzen	AUF	3
Ändern der Einstellungen für ALOG-Dateien während die Datenbank in Benutzung ist		
Neues DAL-Kommando DISPLAY ALOG zum Anzeigen der ALOG-Einstellungen	DBB	4
Neue DAL-Kommandos MODIFY ALOG/MODIFY ALOG-RES und MODIFY-ALOG-SIZE zum Ändern der ALOG-Einstellungen	DBB	4
Änderung der Einschränkungen für die UDS-Online-Utility		
WAIT-FOR-TRANSACTION bietet die Möglichkeit zu warten, bis die gesperrte Quellseite von der sperrenden Transaktion freigegeben wird	SIR	8
Mit SET-RELOCATE-PARAMETERS kann auch für *INDEX-LEVELTABLE-PAGES ein Verhalten für den Fall, dass Seiten gesperrt sind, festgelegt werden (CLASH-HANDLING)	SIR	8
BRENAME mit After-Image-Logging: Die Funktion „Datenbank-Objekte umbenennen (BRENAME, BALTER)“ kann auch bei eingeschaltetem After-Image-Logging ausgeführt werden. Auf diese Weise können Logging-Lücken vermieden werden		
Neues Verhalten beim Umbenennungszyklus	AUF	7
Nach einem Umbenennungsprozess können Datenbank-Aktualisierungen durchgeführt werden	SIR	2
Größe für DBTT-Erweiterungen festlegen		
Neuer Operand EXT des DAL-Kommandos ACT DBTT-INCR	DBB	4
Die Ausgabe von BSTATUS enthält zusätzlich den Wert des EXT-Operanden	SIR	6
Neuer Datentyp FIXED REAL BINARY 63		

Erweiterung der Syntaxdarstellung	ENT	4, 9
Änderung von Meldungen, um den neuen Datentyp zu berücksichtigen	MEL	2, 4, 10
Änderungen in V2.9B		
BINILOAD von CSV-Dateien		
Erklärung, wie BINILOAD CSV-Dateien verarbeitet	AUF	2, 5
Änderungen an den Meldungen, um die neue Funktion zu berücksichtigen	MEL	3
BINILOAD mit Datensätzen variabler Länge		
BINILOAD kann mit Datensätzen variabler Länge arbeiten	AUF	5
Änderungen an den Meldungen, um die neue Funktion zu berücksichtigen	MEL	3
Defaultwerte für neue Felder in BALTER		
Neue BALTER-Anweisung, um Felder mit den angegebenen Werten zu füllen	AUF	6
Änderungen an den Meldungen, um die neue Funktion zu berücksichtigen	MEL	3
Anzahl der Extents in DB-JV		
Datenbank-Jobvariable wurde um die Anzahl der Extents der ALOG-Datei erweitert	DBB	9

Tabelle 2: Änderungen in V2.9B gegenüber V2.9

1.5 Darstellungsmittel

In diesem Abschnitt finden Sie die Erläuterung der Piktogramme für Warnhinweise und Hinweise sowie die Zeichenerklärung der Metasprache, wie sie zur Beschreibung von Syntaxregeln benutzt wird.

1.5.1 Warnhinweise und Hinweise

i	Hinweis auf besonders wichtige Informationen
! **VORSICHT!**	Warnhinweis

1.5.2 Nicht-SDF-Darstellungsmittel

Sprachelement	Erklärung	Beispiel
SCHLÜSSELWORT	Schlüsselwörter sind durch Großbuchstaben mit Unterstreichung dargestellt. Sie müssen mindestens die unterstrichenen Teile des Schlüsselwortes angeben.	DATABASE-KEY MANUAL
WAHLWORT	Wahlwörter sind durch Großbuchstaben ohne Unterstreichung dargestellt. Wenn Sie Wahlwörter weglassen, hat das keinen Einfluss auf die Bedeutung einer Klausel.	NAME IS ALLOWED PAGES
variable	Variable sind mit kursiven Kleinbuchstaben dargestellt. Bei der Benutzung eines Formats, in dem eine Variable erscheint, müssen Sie einen aktuellen Wert an ihre Stelle setzen.	*feldname* *literal-3* *ganzzahl*
{ Entweder | oder }	Genau einen der eingeklammerten Ausdrücke müssen Sie angeben. Die Ausdrücke sind durch einen senkrechten Strich getrennt. Eingerückte Zeilen gehören zum vorhergehenden Ausdruck. Die Klammer geben Sie nicht an.	{ CALC | INDEX } { VALUE IS | VALUES ARE }
[wahlweise]	Den eingeklammerten Ausdruck dürfen Sie weglassen. UDS /SQL benutzt dann Standardwerte. Die Klammern selbst geben Sie nicht an.	[IS *ganzzahl*] [WITHIN *realmname*]
... oder ,...	Den unmittelbar vorstehenden Ausdruck können Sie wahlweise mehrmals wiederholen. Die beiden Sprachelemente unterscheiden Wiederholungen mit Leerzeichen oder mit Komma als Trennzeichen.	*feldname*,... {SEARCH KEY.....}...
..... oder . .	Kennzeichnet Auslassungen aus Gründen der Übersichtlichkeit. Bei der Benutzung der Formate sind diese Auslassungen nicht erlaubt.	SEARCH KEY IS ..... RECORD NAME . .
.	Den Punkt müssen Sie angeben, gefolgt von mindestens einem Leerzeichen. Die Unterstreichung geben Sie nicht an.	SET SECTION. 03 *feldname*..... .
Zwischenraum	Bedeutet, dass Sie mindestens ein Leerzeichen angeben müssen.	USING CALC

Tabelle 3: Zeichen der Metasprache

Alle übrigen Zeichen wie ( ) , . ; „ “ = sind keine Metazeichen:
Sie müssen sie so angeben, wie sie im Format dargestellt sind.

1.5.3 SDF-Syntaxdarstellung

Diese Syntaxbeschreibung basiert auf der SDF-Version 4.7. Die Syntax der SDF-Kommando-/Anweisungssprache wird im Folgenden in 3 Tabellen erklärt.

Tabelle 4 : Metasyntax

In den Kommando-/Anweisungsformaten werden bestimmte Zeichen und Darstellungsformen verwendet, deren Bedeutung in Tabelle 4 erläutert wird.

Tabelle 5 : Datentypen

Variable Operandenwerte werden in SDF durch Datentypen dargestellt. Jeder Datentyp repräsentiert einen bestimmten Wertevorrat. Die Anzahl der Datentypen ist beschränkt auf die in Tabelle 5 beschriebenen Datentypen.

Die Beschreibung der Datentypen gilt für alle Kommandos und Anweisungen. Deshalb werden bei den entsprechenden Operandenbeschreibungen nur noch Abweichungen von Tabelle 5 erläutert.

Tabelle 6 : Zusätze zu Datentypen

Für den Datentyp integer enthält Tabelle 6 außerdem kursiv gesetzte Einheiten, die nicht Bestandteil der Syntax sind. Sie dienen lediglich als Lesehilfe.

Die Beschreibung der Zusätze zu den Datentypen gilt für alle Kommandos und Anweisungen. Deshalb werden bei den entsprechenden Operandenbeschreibungen nur noch Abweichungen von Tabelle 6 erläutert.

Kennzeichnung	Bedeutung	Beispiele
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Schlüsselwörter. Einige Schlüsselwörter beginnen mit *	OPEN DATABASE COPY-NAME = *NONE
=	Das Gleichheitszeichen verbindet einen Operandennamen mit den dazugehörenden Operandenwerten.	CONFIGURATION-NAME = <name 1..8>
< >	Spitze Klammern kennzeichnen Variablen, deren Wertevorrat durch Datentypen und ihre Zusätze beschrieben wird (Tabellen 5 und 6).	DATABASE = <dbname>
Unterstreichung	Der Unterstrich kennzeichnet den Standardwert eines Operanden.	SCHEMA-NAME = *STD
/	Der Schrägstrich trennt alternative Operandenwerte.	CMD = *ALL / <dal-cmd>

(...)	Runde Klammern kennzeichnen Operandenwerte, die eine Struktur einleiten.	*KSET-FORMAT(...)
Einrückung	Die Einrückung kennzeichnet die Abhängigkeit zu dem jeweils übergeordneten Operanden.	USER-GROUP-NAME = *KSET-FORMAT(...) *KSET-FORMAT(...) | HOST = <host>
| | |	Der Strich kennzeichnet zusammengehörende Operanden einer Struktur. Sein Verlauf zeigt Anfang und Ende einer Struktur an. Innerhalb einer Struktur können weitere Strukturen auftreten. Die Anzahl senkrechter Striche vor einem Operanden entspricht der Strukturtiefe.	USER-GROUP-NAME = *ALL-EXCEPT(...) *ALL-EXCEPT(...) | NAME = *KSET-FORMAT(...) | *KSET-FORMAT(...) | | HOST = <host> | | ...
,	Das Komma steht vor weiteren Operanden der gleichen Strukturstufe.	,SPACE = STD
list-poss(n):	Aus den `list-poss` folgenden Operandenwerten kann eine Liste gebildet werden. Ist (n) angegeben, können maximal n Elemente in der Liste vorkommen. Enthält die Liste mehr als ein Element, muss sie in runde Klammern eingeschlossen werden.	NAME = list-poss(30): <subschemaname>

Tabelle 4: Metasyntax

Datentyp	**Zeichenvorrat**	**Besonderheiten**
alog-seq-nr	0..9	1..9 Zeichen
appl	A..Z 0..9 $,#,@ Strukturkennzeichen: Bindestrich	1..8 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen A..Z oder $, #, @ Wenn weniger als 8 Zeichen angegeben werden, wird intern mit Unterstrich auf 8 Zeichen aufgefüllt.
catid	A..Z 0..9	1..4 Zeichen; darf nicht mit der Zeichenfolge PUB beginnen

copyname	A..Z 0..9	1..7 Zeichen beginnend mit A..Z
c-string	EBCDIC-Zeichen	1..4 Zeichen ist in Hochkommata einzuschließen; der Buchstabe C kann vorangestellt werden; Hochkommata innerhalb des c-string müssen verdoppelt werden
csv-dateiname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen ist in Hochkomma einzuschließen
dal-cmd	A..Z 0..9 Bindestrich	1..64 Zeichen
date	0..9 Strukturkennzeichen: Bindestrich	Angabe eines Datums Eingabeformat: jjjj-mm-tt jjjj : Jahr; wahlweise 2- oder 4-stellig mm : Monat tt : Tag
dbname	A..Z 0..9	1..17 Zeichen beginnend mit A..Z
device	A..Z 0..9 $,#,@ Strukturkennzeichen: Bindestrich	5..8 Zeichen beginnend mit A..Z oder 0..9 Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann und die einem im System verfügbaren Gerät entspricht. In der Dialogführung zeigt SDF die zulässigen Operandenwerte an. Hinweise zu möglichen Geräten sind der jeweiligen Operandenbeschreibung zu entnehmen.
host	A..Z 0..9 $,#,@ Strukturkennzeichen: Bindestrich	1..8 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen A..Z oder $, #, @; wenn weniger als 8 Zeichen angegeben werden, wird intern mit Unterstrich auf 8 Zeichen aufgefüllt
integer	0..9,+,-	+ bzw. - kann nur erstes Zeichen sein.
kset	A..Z 0..9 $,#,@ Strukturkennzeichen: Bindestrich	1..8 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen A..Z oder $, #, @; wenn weniger als 8 Zeichen angegeben werden, wird intern mit Unterstrich auf 8 Zeichen aufgefüllt.

name	A..Z 0..9 $,#,@	1..8 Zeichen darf nicht nur aus 0..9 bestehen oder darf nicht mit einer Ziffer beginnen.
realmname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A..Z
realmref	0..9	1..3 Zeichen
recordname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A..Z Bei Satzarten mit Searchkey wird empfohlen, max. 26 Zeichen lange Namen zu verwenden, da ansonsten der implizit gebildete Setname (SYS_...) entsprechend der Begrenzung der Namenslänge von Sets gekürzt wird.
recordref	0..9	1..3 Zeichen
schemaname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A..Z
setname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen A..Z
structured-name	A...Z 0...9 $, #, @ Bindestrich	alphanumerische Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A...Z oder $, #, @
subschemaname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A..Z
time	0..9 Strukturkennzeichen: Doppelpunkt	Angabe einer Tageszeit `{ hh:mm:ss | hh:mm | hh }` hh, mm, ss: bei Stunden, Minuten und Sekunden können führende Nullen weggelassen werden

userid	A..Z 0..9 $,#,@	1..8 Zeichen beginnend mit A..Z oder $,#,@ BPRIVACY: Wenn weniger als 8 Zeichen angegeben werden, wird intern mit Unterstrich auf 8 Zeichen aufgefüllt.
volume	A..Z 0..9 $,#,@	1..6 Zeichen beginnend mit A..Z oder 0..9
x-string	Sedezimal: 00..FF	1..8 Zeichen ist in Hochkommata einzuschließen; der Buchstabe X muss vorangestellt werden; die Anzahl der Zeichen darf ungerade sein.

Tabelle 5: Datentypen

Zusatz	**Bedeutung**
x..y *unit*	beim Datentyp integer: Intervallangabe x Mindestwert, der für integer erlaubt ist. x ist eine ganze Zahl, die mit einem Vorzeichen versehen werden darf. y Maximalwert, der für integer erlaubt ist. y ist eine ganze Zahl, die mit einem Vorzeichen versehen werden darf. *unit* nur bei integer: zusätzliche Einheiten. Folgende Angaben werden verwendet: *Mbyte* *Kbyte* *seconds*

Tabelle 6: Zusätze zu Datentypen

1.6 Beispieldatenbanken

Zu den Beispielen in diesem Handbuch wurde eine Datenbankkonfiguration aufgebaut, die aus den folgenden vier Datenbanken besteht:

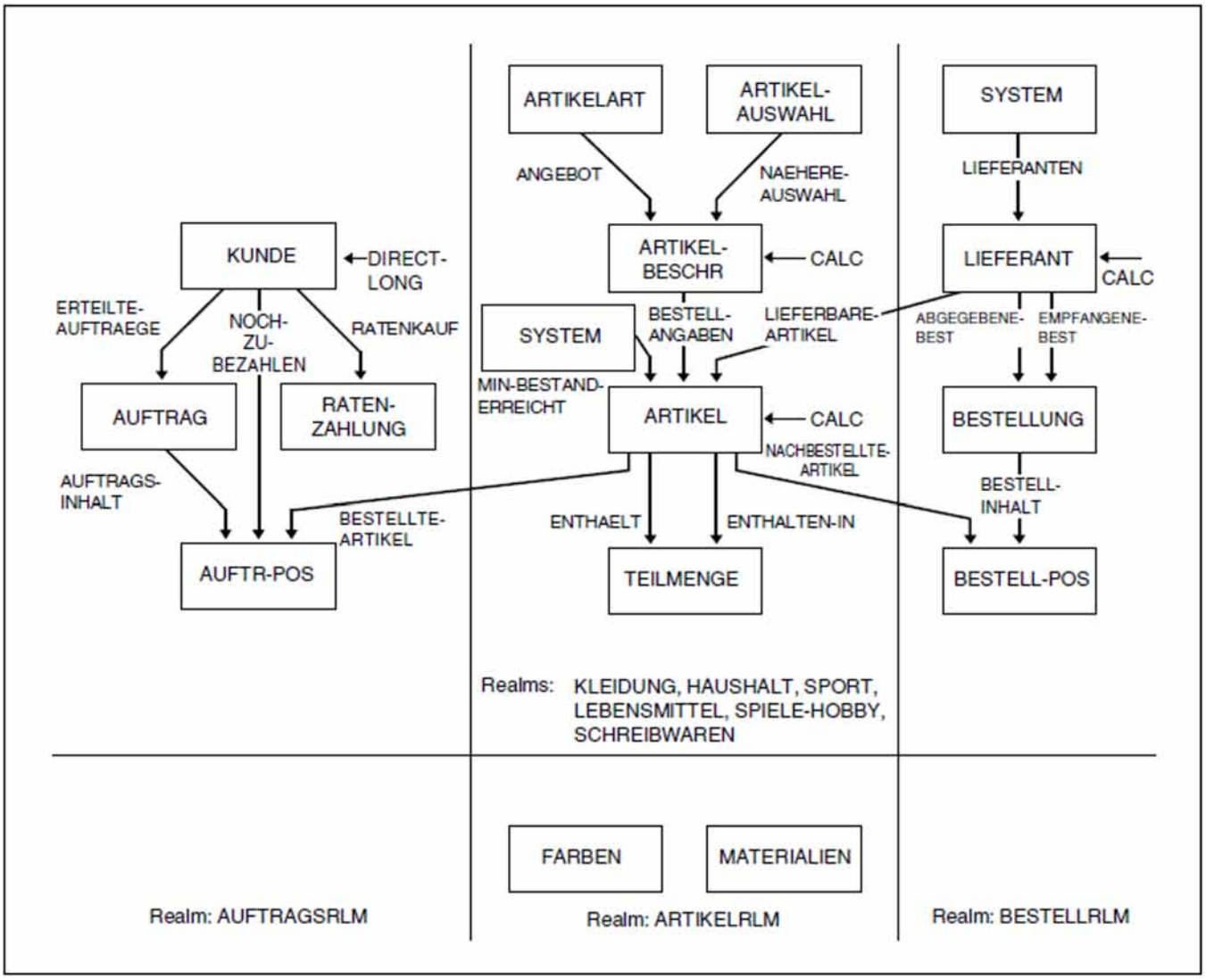

Bild 1: Datenbank VERSAND mit dem Schemanamen ARTIKELVERSAND

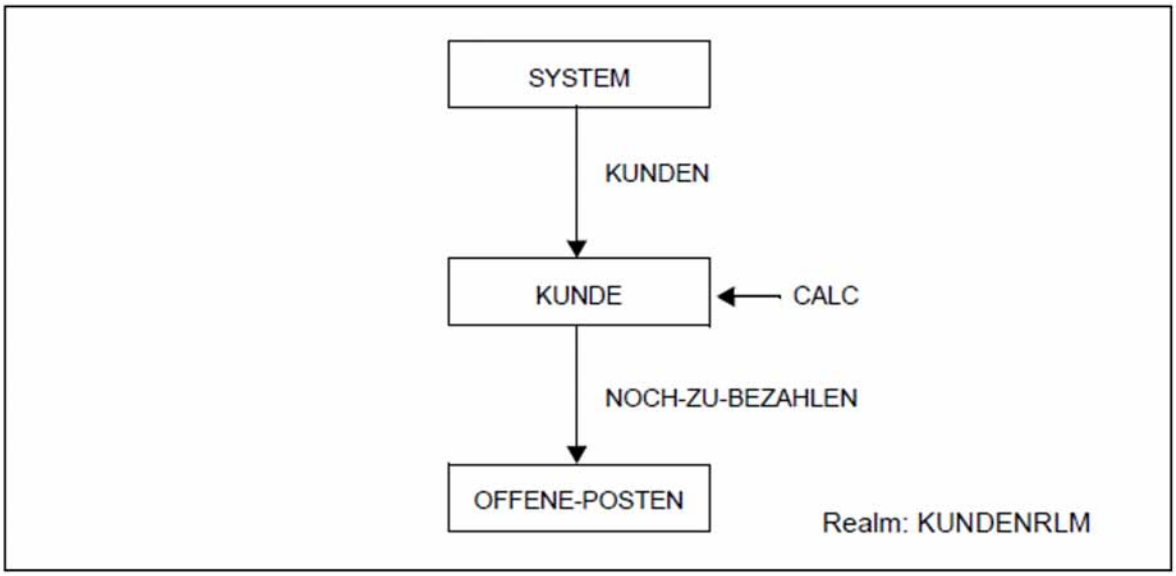

Bild 2: Datenbank KUNDEN mit dem Schemanamen KUNDENDATEI

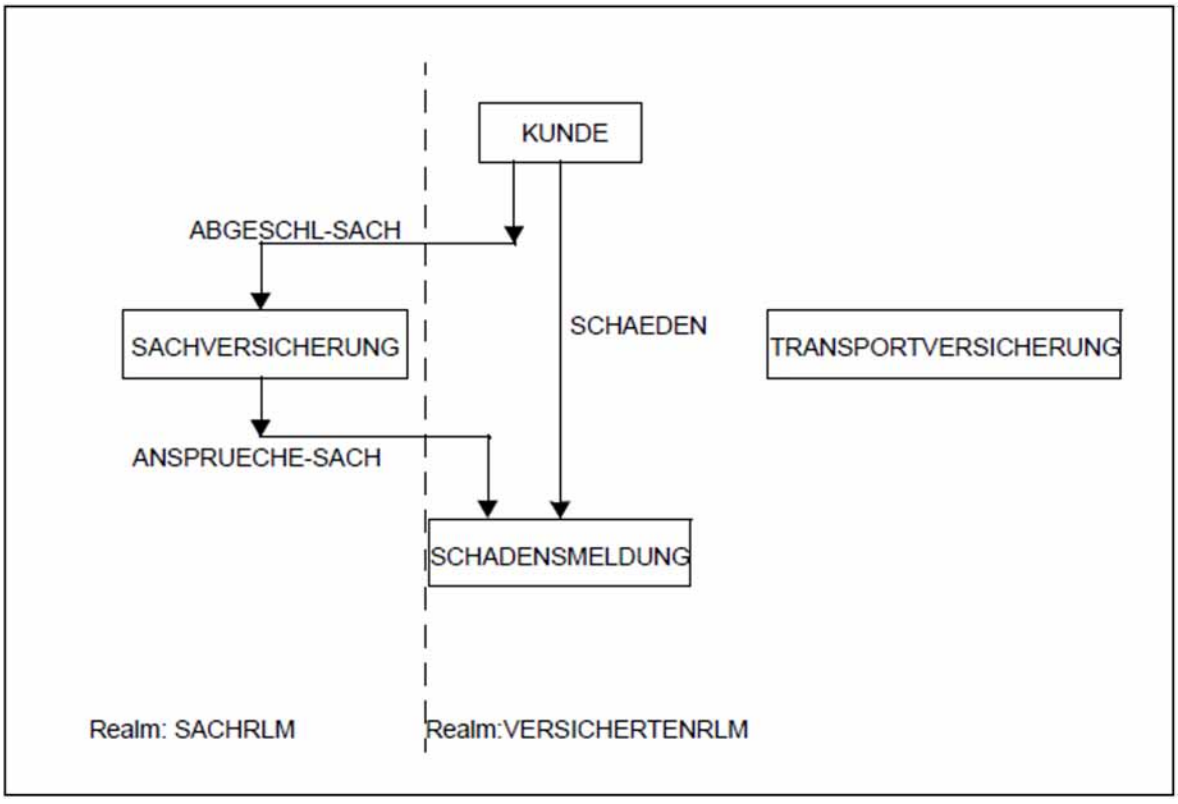

Bild 3: Datenbank VERS

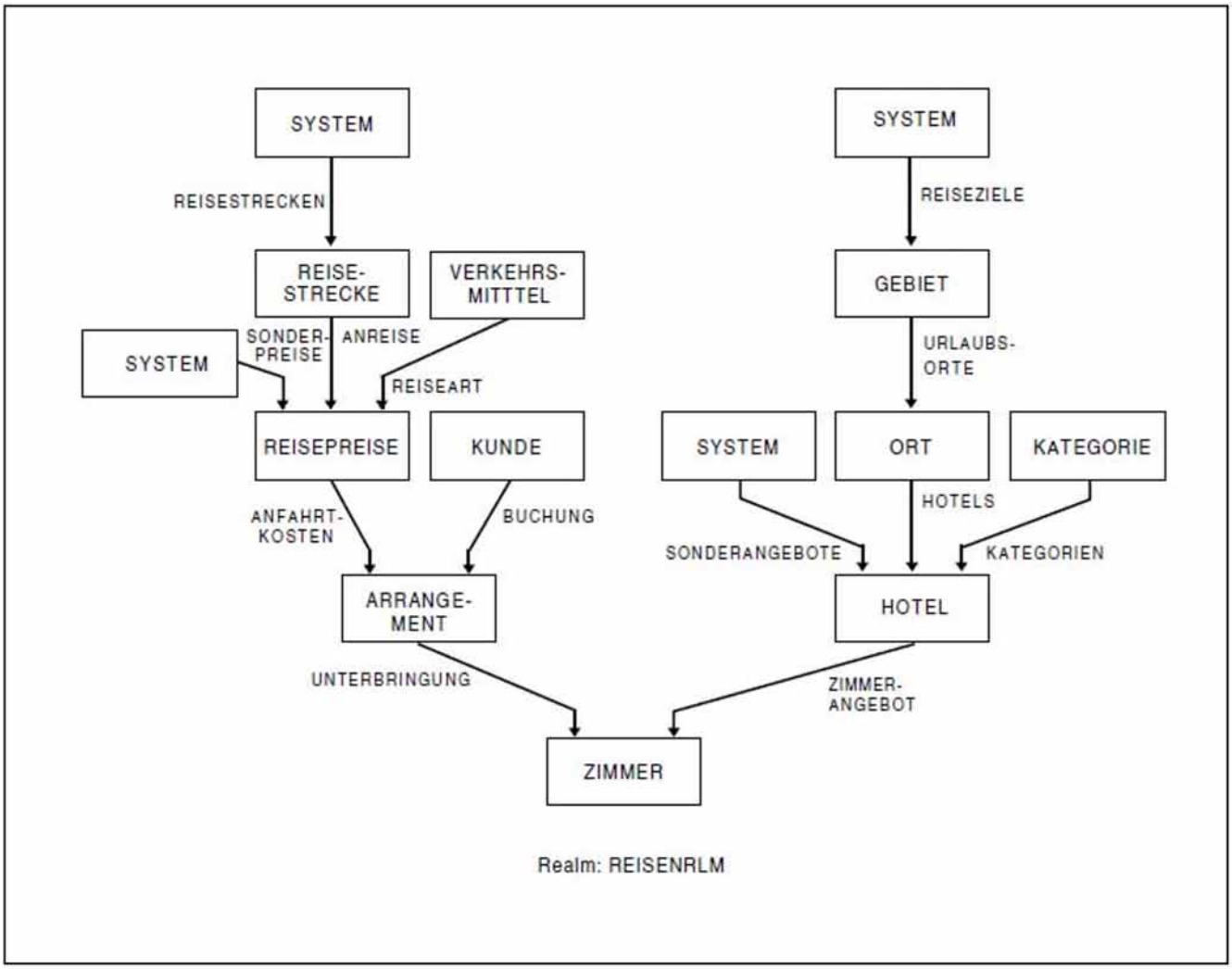

Bild 4: Datenbank REISEN mit dem Schemanamen REISEBUERO

2 UDS/SQL im Überblick

Dieses Kapitel führt in die Grundbegriffe von UDS/SQL ein und gibt einen Überblick über die Realms und Dateien einer UDS/SQL-Datenbank. Anschließend stellt es die Programme des Datenbanksystems UDS/SQL vor.

2.1 Grundbegriffe des Datenbanksystems UDS/SQL

UDS/SQL-Datenbank

In einer UDS/SQL-Datenbank sind große Mengen aufeinander bezogener Daten zusammengefasst. Die Daten werden in der Datenbank so gespeichert, dass sie unabhängig von der Programmierung sind und von verschiedenen Programmen optimal verwendet werden können, trotzdem aber so wenig Redundanzen wie möglich aufweisen. Kontrolliert werden neue Daten zur Datenbank hinzugefügt, existierende Daten wiedergewonnen, geändert oder gelöscht.
Bei UDS/SQL können mehrere Datenbanken zu einem gemeinsam prozessierbaren Multi-DB-System zusammengefasst werden, siehe auch Abschnitt „Datenbankkonfiguration".

Datenbanksystem

Die Gesamtheit aller Programme, die für den Aufbau und die Verwaltung der Datenbestände sowie für die Wiedergewinnung und Sicherung der Daten erforderlich sind, bezeichnet man als Datenbanksystem.

Database Handler (DBH)

Den Zugriff auf die Datenbank steuert die zentrale Komponente von UDS/SQL, der Database Handler (DBH), der eine Datenbank (Mono-DB-Betrieb) oder auch mehrere Datenbanken gemeinsam (Multi-DB-Betrieb) prozessieren kann.
Er wird angeboten in den folgenden beiden Varianten:

- independent DBH
- linked-in DBH

Beim **independent DBH** steuern die folgenden Module den Datenbankbetrieb:

- UDSSQL
- UDSSUB
- UDSCT bei UDS-D

Jedes Modul läuft als eigene Task ab. Zusammen bilden sie die Taskfamilie des independent DBH.

Die Module haben folgende Aufgaben:

UDSSQL	Mastertask; UDSSQL führt die Kommunikation mit dem Datenbankadministrator und sorgt für das Einleiten, Überwachen und Beenden der Session.
UDSSUB	Servertask (mehrfach ladbar); UDSSUB erhält die Anforderungen der Anwenderprogramme zur Bearbeitung und gibt die Ergebnisse an das jeweilige Anwenderprogramm zurück.
UDSCT	UDS-D-Task; UDSCT übernimmt die Kommunikationsaufgaben, die nötig sind, um DML-Anweisungen von entfernten Anwenderprogrammen zu bearbeiten.

Der **linked-in DBH** ist kein selbstständiges Programm, sondern wird in das jeweilige Anwenderprogramm eingebunden oder zur Laufzeit nachgeladen und läuft als Teil dieses Programms. Datenbankänderungen können von einem solchen Anwenderprogramm nur dann ausgeführt werden, wenn es als einziges (EXCLUSIVE) mit der

Datenbank arbeitet. RETRIEVAL-Zugriff durch mehrere linked-in DBHs ist möglich. Da die Task-Kommunikation, die beim independent DBH erforderlich ist, hier wegfällt, kann besonders bei sequenzieller Verarbeitung eine Laufzeitverbesserung erzielt werden.

Der linked-in DBH ist, ebenso wie der independent DBH, multi-DB-fähig. Der linked-in DBH kann keine SQL-Anweisungen bearbeiten.

Session

Eine Session ist der Zeitraum, in dem ein oder mehrere Anwender mit der (den) Datenbank(en) arbeiten können. Sie beginnt mit dem Laden des DBH (independent oder linked-in) und endet mit der Meldung „`NORMAL SYSTEM TERMINATION`“. Innerhalb der Session ist die Datenbankkonfiguration durch Parameter festgelegt, die der Datenbankadministrator beim Laden des DBH oder per DAL-Kommando angibt.

Datenbankkonfiguration

Beim Starten der Session legt der Datenbankadministrator mit den Ladeparametern des DBH fest, welche Datenbanken zunächst an der Session teilnehmen. Die Datenbanken einer Session und die Umgebung, in der die Session abläuft, nennt man Datenbankkonfiguration.

Jede Datenbankkonfiguration erhält vom Administrator einen Namen. Die Konfigurationsdaten werden für die Dauer der Session in der Session-Log-File (SLF) hinterlegt, damit bei einem eventuellen Wiederanlauf die aktuelle Datenbankkonfiguration wiederhergestellt werden kann.

Transaktion

Jedes Datenbank-Anwenderprogramm muss beim DBH eine Transaktion eröffnen, wenn es mit einer UDS/SQL-Datenbank kommunizieren will - gleichgültig, ob über den linked-in oder den independent DBH.

Eine Transaktion (TA) ist eine logisch zusammengehörige Folge von DML-Anweisungen, die entweder vollständig oder gar nicht ausgeführt wird. Beispielsweise beginnt eine Transaktion in einem COBOL-DML-Programm mit der DML-Anweisung READY und endet mit FINISH.
Im Mono-DB-Betrieb eröffnen Sie mit jeder READY-Anweisung eine Transaktion und gleichzeitig die Datenbank; das Anwenderprogramm kann nun beliebig oft per DML-Anweisung auf die eröffnete Datenbank zugreifen. Im Multi-DB-Betrieb müssen Sie jede Datenbank mit einer READY-Anweisung eröffnen. Deshalb kann es im Multi-DB-Betrieb innerhalb einer Transaktion mehrere READY-Anweisungen geben, wobei die erste READY-Anweisung die Transaktion eröffnet.
Beendet das COBOL-DML-Anwenderprogramm die Transaktion mit der DML-Anweisung FINISH, so kann es erst dann wieder auf die Datenbank(en) zugreifen, wenn gilt:

- Im Mono-DB-Betrieb hat das Anwenderprogramm mit einem erneuten READY eine neue Transaktion und damit auch die Datenbank eröffnet.
- Im Multi-DB-Betrieb hat das Anwenderprogramm mit einem erneuten READY eine neue Transaktion und damit auch eine Datenbank eröffnet. Mit weiteren READY-Anweisungen innerhalb derselben Transaktion wurden ggf. weitere Datenbanken eröffnet.

Auch ein SQL-Programm kann nur innerhalb von Transaktionen auf UDS/SQL-Datenbanken zugreifen. In SQL-Programmen beginnt eine Transaktion mit der ersten von COMMIT WORK verschiedenen SQL-Anweisung und endet mit der SQL-Anweisung COMMIT WORK.

Vorgang

In einer SQL-Anwendung werden SQL-spezifische Verwaltungsdaten über Transaktionsgrenzen hinweg aufbewahrt. Eine solche Verwaltungseinheit wird als Vorgang bezeichnet.

2.2 Dateien und Realms einer UDS/SQL-Datenbank

Eine UDS/SQL-Datenbank besteht aus der Benutzerdatenbank und der Compilerdatenbank und wird aus mehreren Datenbankbereichen (Realms) gebildet.

Die **Benutzerdatenbank** beinhaltet alle Realms und Dateien, die der Anwender benötigt, um Daten in die Datenbank zu speichern und wiederzugewinnen. Dazu gehören:

- die Benutzerrealms
- das Database Directory (DBDIR)
- die Modulbibliothek für Hashroutinen (HASHLIB)

Die **Compilerdatenbank** enthält die übersetzten Schema- und Subschema-Beschreibungen; sie wird vom DDL-Compiler und vom COBOL-Compiler benötigt. Zu ihr gehören:

- das Database Directory (DBDIR)
- der Database Compiler Realm (DBCOM)
- das COBOL Subschema Directory (COSSD)

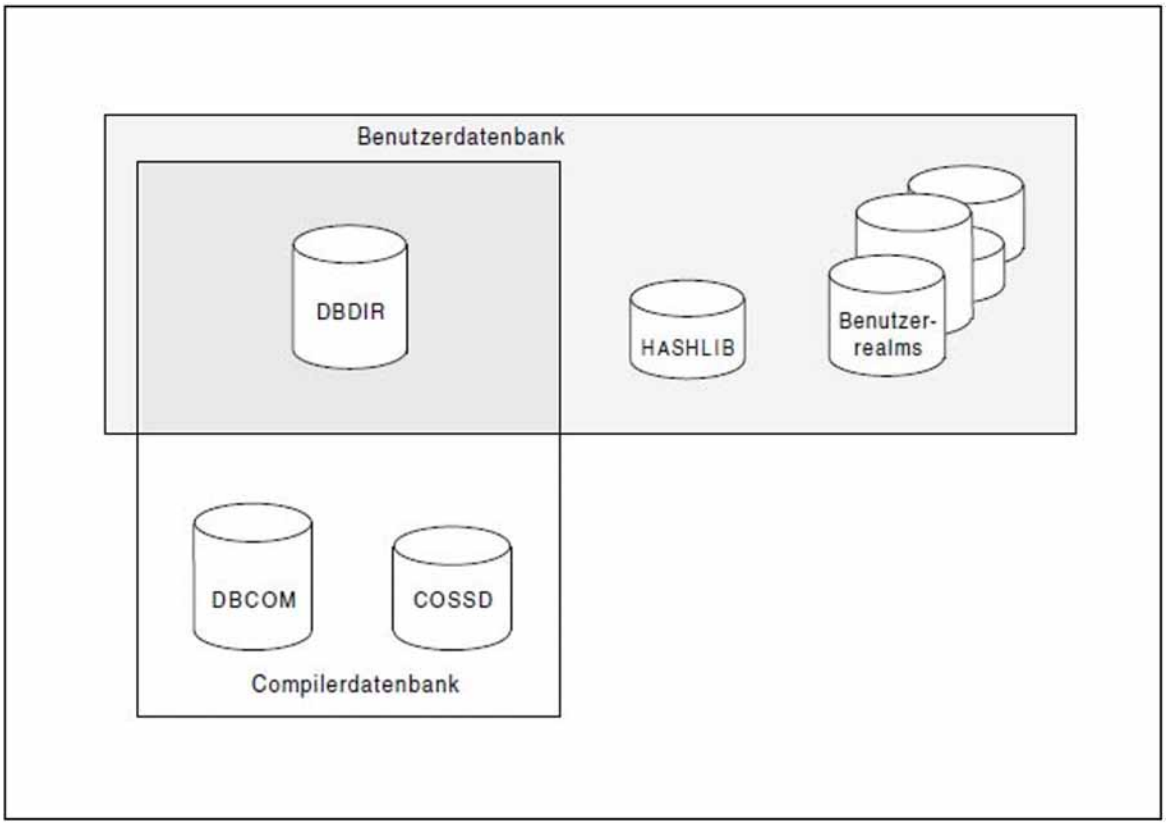

Bild 5: Die UDS/SQL-Datenbank

Außer den Dateien der Benutzerdatenbank und der Compilerdatenbank werden noch einige Dateien für den Verbindungsaufbau mit der Datenbank und die Datensicherung benötigt. Die folgende Zusammenstellung zeigt eine komplette Übersicht über die Dateien einer UDS/SQL-Datenbank:

Datenbankrealms

dbname.realmname	Original-Benutzerrealm(s)
dbname .DBDIR	Database Directory
dbname .DBCOM	Database Compiler Realm

Dateien für den Datenbankbetrieb

dbname	Verbindungsdatei zur Datenbank bei Mono-DB-Betrieb
dbname .COSSD	COBOL Subschema Directory
dbname .HASHLIB	Bibliothek zur Aufnahme der Hashroutinen
konfigurationsname	Verbindungsdatei zur Datenbankkonfiguration
konfname .DBSTAT	DB-Status-Datei
konfname.DBSTAT.SAVE	Duplikat der DB-Status-Datei
konfname .SLF	Session-Log-File (SLF)
konfname.TEMPO.*nnn*	temporäre Benutzerdatei
UDS.ENTER.*tsn*.ST0*nn*	ENTER-Dateien zum Starten der Servertasks.

Dateien zur Sicherung der Datenbank

Schattendatenbank:

- *dbname*.DBDIR.*copyname*
- *dbname*.DBCOM.*copyname*
- *dbname*.COSSD.*copyname*
- *dbname*.HASHLIB.*copyname*
- *dbname.realmname.copyname*

ALOG-Datei:

- *dbname*.A.*folgenr*

RLOG-Dateien:

- *konfname*.RLOG.*rlogzeitstempel*.1
- *konfname*.RLOG.*rlogzeitstempel*.2
- *konfname*.RLOG.*rlogzeitstempel*.1.SAVE
- *konfname*.RLOG.*rlogzeitstempel*.2.SAVE

Dateien der mit BPGSIZE umgestellten Datenbank

- *dbname*.DBDIR.NEW
- *dbname*.DBCOM.NEW
- *dbname*.COSSD.NEW (falls mit BPGSIZE auf größeres Seitenformat umgestellt wird)
- *dbname*.*realmname*.NEW

Syntaxregeln

dbname

Name der Datenbank mit max. 17 Zeichen Länge. Er kommt als Teilqualifikation in fast allen Dateinamen der Datenbankrealms und Datenbankdateien vor. Er muss folgenden Regeln entsprechen:

- *dbname* muss bei allen Dateien der Datenbank gleich sein.
- *dbname* darf weder Sonderzeichen noch Leerzeichen enthalten und das erste Zeichen muss ein Buchstabe sein.
- :*catid*:$*userid*.*dbname*.*realmname*.*copyname* darf insgesamt maximal 54 Zeichen lang sein.

Bei UDS-D gilt:

- *dbname* muss netzweit eindeutig sein.

konfigurationsname

frei wählbarer Name der Datenbankkonfiguration; kann im Mono-DB-Betrieb mit *ddbname* identisch sein. Es gelten folgende Regeln:

- *konfigurationsname* darf max. 41 Zeichen lang sein;
- alle *konfigurationsnamen* einer BS2000-Session müssen in den ersten sieben Zeichen netzweit eindeutig sein;
- *konfigurationsname* darf in den ersten acht Stellen kein Sonderzeichen enthalten.

konfname

die ersten acht Zeichen des Namens der Datenbankkonfiguration, den der DB-Administrator beim Starten der Session festlegt

Bei UDS-D gilt:

- *konfname* muss in den ersten sieben Zeichen eindeutig sein.

nnn Nummer der Transaktion, der die Datei zugeordnet ist

nn Nummer der Servertask

pool Name des Common Memory Pools

tsn Prozessfolgenummer der Mastertask

realmname

Name eines Realm der Datenbank;

bei Benutzerrealms: mit der AREA-Klausel der Schema-DDL definierter Name des Realm

beim Database Directory: DBDIR

beim Database Compiler Realm: DBCOM

copyname

Suffix für die Schattendatenbank. *copyname* besteht aus maximal sieben Zeichen

folgenr

Neunstellige Folgenummer, die jeder ALOG-Datei zugeordnet ist

rlogzeitstempel

Zeitpunkt, zu dem die jeweilige RLOG-Datei eröffnet wurde

Im Folgenden sind alle Dateien und Realms einer UDS/SQL-Datenbank kurz beschrieben.

Datenbankrealms

- Benutzerrealms (*dbname*.*realmname*)

  Zur Aufnahme der Daten müssen Sie die Datenbankrealms einrichten, die in der Schema-DDL mit der AREA-Klausel definiert wurden.

- Database Directory (*dbname*.DBDIR)

  Das Database Directory (DBDIR) enthält die vollständige Schema-Beschreibung, alle Subschema-Beschreibungen und Informationen über die Zugriffsberechtigungen. Außerdem enthält es Informationen darüber, ob AFIM-Logging ein- oder ausgeschaltet ist, Realms zu- oder abgeschaltet sind oder im Rahmen einer Umstrukturierung gelöscht wurden.
  Der Database Handler benötigt diese Informationen, um für den Anwender die Datenbankzugriffe innerhalb des Bereichs des verwendeten Subschemas zu erledigen.

- Database Compiler Realm (*dbname*.DBCOM)

  Der Database Compiler Realm (DBCOM) speichert Einzelheiten über die Realms, Sätze und Sets, die der Anwender in der Schema-DDL und der Subschema-DDL definiert hat. Der DBCOM wird nur zum Übersetzen der Schema-DDL und der Subschema-DDL sowie beim Aufbauen des DBDIR und des COSSD benötigt.

Dateien für den Datenbankbetrieb

- Verbindungsdatei (*dbname*)

  Leere Verbindungsdatei zur Datenbank bei Mono-DB-Betrieb

- COBOL Subschema Directory (*dbname*.COSSD)

  In das COBOL Subschema Directory (COSSD) schreibt der DDL-Compiler nach der Übersetzung des Subschemas Informationen über das übersetzte Subschema. Diese Informationen benötigt der COBOL-Compiler zur Übersetzung der DB-Anwenderprogramme. Der COBOL-Compiler entnimmt dem COSSD die Datenstruktur des Subschemas und prüft mit Hilfe einer Tabelle im COSSD die Zulässigkeit der DML-Befehle.

- Modulbibliothek für Hashroutinen (*dbname*.HASHLIB)

  Die Modulbibliothek *dbname*.HASHLIB speichert die Hashroutinen für die Datenbank.

- Verbindungsdatei (*konfigurationsname*)

  Leere Verbindungsdatei zur DB-Konfiguration bei Multi-DB-Betrieb.

- DB-Status-Datei
  (*konfname*.DBSTAT)
  (*konfname*.DBSTAT.SAVE)

  Die DB-Status-Datei wird von *open*UTM beim Wiederanlauf benötigt; sie enthält Informationen über die letzten zurückgesetzten Transaktionen einer jeden
  UDS/SQL-/*open*UTM-Anwendung. Die DB-Status-Datei wird aus Sicherheitsgründen doppelt geführt.

  Bei UDS-D gilt:

  Bei verteilter Verarbeitung mit UDS-D kann die Datei außerdem Informationen enthalten, die beim Beenden der Transaktion abgelegt werden.

- Session-Log-File (*konfname*.SLF)

  Die Session-Log-File (SLF) benötigt der DBH bei einem Wiederanlauf. Sie enthält Informationen über die Datenbanken, die an die Konfiguration angeschlossen sind und die aktuellen Werte der Ladeparameter des DBH.

- temporäre Benutzerdatei (*konfname*.TEMPO.*nnn*)

  Wenn für (mindestens) eine der Datenbanken der Konfiguration ein Temporärer Realm deklariert ist, richtet der DBH für jede Mainreference (parallel offene Transaktion) eine temporäre Datei ein. Diese Datei speichert temporäre Informationen.

  nnn Nummer der Mainreference

- ENTER-Dateien (UDS.ENTER.*tsn*.ST0*nn*)

  Die Mastertask erzeugt eine oder mehrere ENTER-Dateien für Servertasks. (UDS.ENTER.*tsn*.ST0*nn*)

  Die Mastertask startet diese ENTER-Prozesse mit ENTER-Kommandos. Bei normaler Beendigung der Session löscht die Mastertask alle ENTER-Dateien.

Dateien zur Datensicherung

- Schattendatenbank

  (*dbname*.DBDIR.*copyname*)
  (*dbname*.DBCOM.*copyname*)
  (*dbname*.COSSD.*copyname*)
  (*dbname*.HASHLIB.*copyname*)
  (*dbname.realmname.copyname*)

  Mit dem COPY-FILE-Kommando können Sie die Realms und Dateien der Datenbank kopieren. Mit dem MODIFY-FILE-ATTRIBUTES-Kommandos können Sie die Datenbank umbenennen. Die Datenbank kann auch mit dem BS2000-Dienstprogramm ARCHIVE gesichert werden.

- ALOG-Dateien (*dbname*.A.*folgenr*)

  In die ALOG-Dateien trägt der DBH oder ein änderndes Dienstprogramm jede Änderung einer Seite ein und zwar jeweils den Stand *nach* der Veränderung (After-Image). Mit Hilfe der ALOG-Dateien lassen sich somit Änderungen nachvollziehen.
  Die Schattendatenbank hat keine ALOG-Dateien, die Änderungen lassen sich aber mit den ALOG-Dateien der Originaldatenbank in die Schattendatenbank nachfahren.

- RLOG-Dateien

  (*konfname*.RLOG.*rlogzeitstempel*.1)
  (*konfname*.RLOG.*rlogzeitstempel*.2)
  (*konfname*.RLOG.*rlogzeitstempel*.1.SAVE)
  (*konfname*.RLOG.*rlogzeitstempel*.2.SAVE)

  In den RLOG-Dateien protokolliert der DBH bei Bedarf für eventuell erforderliche Rollbacks oder Warmstarts eine zu ändernde Information (Datum) **vor** ihrer Änderung, genannt Before-Image (BFIM), bzw. **nach** ihrer Änderung, genannt After-Image (AFIM).

Maximalgröße für UDS/SQL-Dateien

UDS/SQL kann maximal 16777214 Datenbankseiten in einem Realm verwalten. Daraus ergeben sich folgende Grenzwerte für die maximalen Dateigrößen:

Datenbankseitengröße	Maximale Dateigröße in PAM-Seiten
2 KB	16777214
4 KB	33554428
8 KB	67108856

Tabelle 7: Grenzwerte für Dateigrößen

UDS/SQL unterstützt als LARGE FILE-Dateien Realmdateien, Tempodateien, Loggingdateien und Arbeitsdateien der Dienstprogramme. Dateien, die von ihrer Natur her nur begrenzten Umfang annehmen, werden von UDS/SQL nicht mit LARGE FILE-Eigenschaft bearbeitet (DBSTAT, COSSD, HASHLIB, Parameterdateien). Auch die Ausgabedateien des Monitors haben nicht die LARGE FILE-Eigenschaft.

Dateien, die von den UDS/SQL-Dienstprogrammen als Hilfsdateien angelegt werden, werden nicht mit BLKCNTRL=PAMKEY angelegt, damit sie nicht implizit die Möglichkeit zur Nutzung als LARGE FILE verlieren.

Zur Nutzung der LARGE FILE-Eigenschaft müssen im System folgende Voraussetzungen erfüllt sein:

- Large Files können nur auf Pubsets mit der Eigenschaft LARGE-FILES-ALLOWED genutzt werden.
- Large Files können im HOME-Pubset nicht genutzt werden.
- Large Files können nicht mit BLKCTRL=PAMKEY genutzt werden.

Kennwörter für UDS/SQL-Dateien

UDS/SQL schützt die automatisch eingerichteten Dateien mit dem Standard-Kennwort:
C’UDS'BLANK''.
Ausnahme: RLOG-Datei. Das Kennwort für die RLOG-Datei wird automatisch vergeben und setzt sich aus Teilen des RLOG-Zeitstempels zusammen. Das Kennwort kann nur in der Systemkennung ($TSOS) unter Umgehung des Kennwortschutzes gelöscht werden.

2.3 Programme des Datenbanksystems UDS/SQL

Zum Gesamtsystem UDS/SQL (BS2000) gehören eine Reihe von Programmen, die zum Einrichten und Warten der Datenbank sowie zur Kommunikation mit der Datenbank erforderlich sind.

Die Funktionen dieser Programme sind im Folgenden kurz erläutert:

Datenbank einrichten	**Programmlauf vorbereiten**	**Datenbank laden und entladen**	**Session überwachen**	**mit der Datenbank arbeiten**	**DML-Funktionen testen**
BCREATE DDL SSL BGSIA BFORMAT BGSSIA BPRIVACY OPRIVACY	BCALLSI	BINILOAD BOUTLOAD	UDSMON	IQS (gehört nicht zum Lieferumfang von UDS/SQL)	DMLTEST
D a t e n b a n k w a r t u n g					
Informationen ausgeben	**Datenbank reorganisieren**	**Datenbank umstrukturieren/ Datenbankobjekte umbenennen**	**Datenbank wiederherstellen**	**Datenbank prüfen**	**Datenbank umstellen**
BPSIA BPSQLSIA BSTATUS BPRECORD	BREORG BMODTT ONLUTIL	BCHANGE BRENAME BALTER	BMEND	BCHECK	BPGSIZE BTRANS24
D a t e n b a n k b e t r i e b					
		UDS/SQL administrieren			
		UDSADM			

Tabelle 8: Programmübersicht

Datenbank einrichten

BCREATE	formatiert das DBDIR und den DBCOM.
DDL	DDL-Compiler; übersetzt die Schema-DDL und die Subschema-DDL und baut DBCOM sowie COSSD auf.
SSL	SSL-Compiler; übersetzt die SSL und modifiziert Daten im DBCOM.
BGSIA	baut die Schema Information Area (SIA) auf und speichert sie im DBDIR.
BFORMAT	formatiert die Benutzerrealms der Datenbank und modifiziert die SIA.
BGSSIA	baut die Subschema Information Area (SSIA) auf und speichert sie im DBDIR.

BPRIVACY bzw. OPRIVACY

trägt die Zugriffsberechtigungen der Anwender im DBDIR ein.

Programmlauf vorbereiten

BCALLSI nur erforderlich, wenn CALL-DML eingesetzt wird!
BCALLSI stellt für Anwender der CALL-DML Subschema-Informationen bereit.

Datenbank laden

BINILOAD lädt schnell große Datenmengen einer Satzart in eine Datenbank.

Datenbank entladen

BOUTLOAD kopiert, löscht oder entlädt Satzarten einer Datenbank.

Session überwachen

UDSMON gibt im laufenden Datenbankbetrieb die UDS/SQL-Betriebswerte aus.

DML-Funktionen testen

DMLTEST testet einzelne DML-Funktionen im Dialog und in Prozeduren.

Informationen über die Datenbank ausgeben

BPSIA druckt einen Überblick aus über die wichtigsten Informationen des Schemas oder eines Subschemas der Datenbank.

BPSQLSIA druckt die relationale Schema-Information eines bestehenden UDS/SQL-Subschemas aus, das gemäß dem CODASYL-Konzept definiert wurde. Die relationale Schema-Information dient als Programmierunterlage für den SQL-Anwender.

BSTATUS erstellt Statistiken über die Speicherplatzbelegung in den Realms der Datenbank.

BPRECORD gibt den Inhalt von Datenbankrealms aus.

Datenbank reorganisieren

BREORG vergrößert und verkleinert Realms der Datenbank, erhöht und reduziert die zulässige Satzanzahl einer Satzart und reorganisiert Tabellen und Hashbereiche.

BMODTT steuert die Wiederverwendung frei gewordener Database-Key-Werte sowie die Freiplatzsuche durch den DBH.

ONLUTIL verlagert Sätze innerhalb einer Datenbank und verändert Einstellungen zu einer Datenbank.

Datenbank umstrukturieren/umbenennen

BCHANGE bereitet DBDIR, DBCOM und COSSD für die Umstrukturierung vor.

BRENAME bereitet DBDIR, DBCOM und COSSD für die Umbenennung vor.

BALTER führt die Umstrukturierung/Umbenennung der vorhandenen Datenbank entsprechend der neuen Schema-Beschreibung aus.

Datenbank wiederherstellen

BMEND richtet ALOG-Dateien ein und bietet Funktionen zum Wiederherstellen einer zerstörten Datenbank sowie auch zur Informationsausgabe über den Zustand nachzufahrender Realms und ALOG-Dateien.

Datenbank prüfen

BCHECK prüft, ob die physischen Strukturen einer Datenbank korrekt sind; es kann im Rahmen der Datensicherung eingesetzt werden, sodass Inkonsistenzen in der Datenbank frühzeitig erkannt und behoben werden können.

Datenbank umstellen

BPGSIZE legt ein neues Seitenformat für die Datenbank fest (Datenbank umstellen).Bei der Umstellung erzeugt BPGSIZE wahlweise

- eine Kopie der Datenbank mit größerer Seitenlänge.
- eine Kopie der Datenbank mit unveränderter Seitenlänge. Die Realms der umgestellten Datenbank haben in der Regel einen geringeren Speicherplatzbedarf.

BTRANS24 setzt Datenbanken der Version UDS/SQL V2.0 bis V2.3 für den Einsatz ab UDS/SQL V2.4 um.

UDS/SQL administrieren

UDSADM Mit dem Programm UDSADM kann eine UDS/SQL-Konfiguration administriert werden.

Einige Dienstprogramme, die auf UDS/SQL-Datenbanken zugreifen, benötigen zum Ablaufen den DBH; diese Programme laden den linked-in DBH nach mit den Standardwerten der Ladeparameter. Alle diese Programme arbeiten nur mit einer Datenbank.

Folgende Tabelle zeigt, welche Programme den linked-in DBH nachladen und welche während des Programmablaufs Dateien der Compilerdatenbank benötigen:

UDS/SQL- Programm	lädt linked-in DBH nach	Zugriff auf		
		DBDIR	DBCOM	COSSD
BALTER	-	X	X	-
BCALLSI	-	-	-	X
BCHANGE	-	X	X	X
BCHECK	-	X	-	-
BCREATE	-	X	X	-
BFORMAT	-	X	-	-
BGSIA	X	X	X	-
BGSSIA	X	X	X	-
BINILOAD	-	X	X [1]	-
BMEND	-	X	X	-

BMODTT	-	X	-	-
BOUTLOAD	-	X	X [2]	-
BPGSIZE	-	X	X	X
BPRECORD	-	X	-	-
BPRIVACY	X	X	X [3]	-
BPSIA	-	X	-	-
BPSQLSIA	-	X	-	X
BRENAME	-	X	X	X
BREORG	-	X	-	-
BSTATUS	-	X	-	-
BTRANS24	-	X	X	-
COBOL-Compiler	-	-	-	X
DDL-Compiler	X	X	X	X
DMLTEST	X[4]	X	-	-
ONLINE-PRIVACY	-	X	X [3]	-
ONLINE-UTILITY	X[4]	X	-	-
SSL-Compiler	X	X	X	-
UDSADM	-	-	-	-
UDSMON	-	-	-	-

Tabelle 9: Übersicht über die Programme des Datenbanksystems UDS/SQL

1	DBCOM wird verwendet, wenn die Eingabedatei im CSV-Format ist oder wenn der Satztyp ein variables Feld enthält
2	DBCOM wird verwendet, wenn CSV-OUTPUT = *YES angegeben wird (UDS/SQL V2.8 und V2.9 sind ebenfalls betroffen)
3	DBCOM wird nicht gelesen, muss aber vorhanden sein
4	Optional

2.3.1 START-Kommandos der UDS/SQL-Programme

Die folgende Tabelle zeigt die START-Kommandos und deren Alias-Namen, mit denen Sie die angegebenen UDS /SQL-Programme aufrufen können.

Folgende Voraussetzungen müssen erfüllt sein:

- UDS/SQL muss mit IMON installiert sein und
- die SDF-Systemsyntaxdatei muss aktiviert sein.

Programm	**START-Kommando**	**Alias-Namen**
UDSSQL	START-UDS-DBH	UDS, SYSINT
BALTER	START-UDS-BALTER	BALTER
BCALLSI	START-UDS-BCALLSI	BCALLSI
BCHANGE	START-UDS-BCHANGE	BCHANGE
BRENAME	START-UDS-BRENAME	BRENAME
BCHECK	START-UDS-BCHECK	BCHECK
BCREATE	START-UDS-BCREATE	BCREATE
BFORMAT	START-UDS-BFORMAT	BFORMAT
BGSIA	START-UDS-BGSIA	BGSIA
BGSSIA	START-UDS-BGSSIA	BGSSIA
BINILOAD	START-UDS-BINILOAD	BINILOAD
BMEND	START-UDS-BMEND	BMEND, START-UDS-REPAIR
BMODTT	START-UDS-BMODTT	BMODTT
BOUTLOAD	START-UDS-BOUTLOAD	BOUTLOAD, START-UDS-OUTLOAD
BPGSIZE	START-UDS-BPGSIZE	BPGSIZE START-UDS-PAGE-RESIZING
BPRECORD	START-UDS-BPRECORD	BPRECORD
BPRIVACY	START-UDS-BPRIVACY	BPRIVACY START-UDS-AUTHORIZATION
BPSIA	START-UDS-BPSIA	BPSIA
BPSQLSIA	START-UDS-BPSQLSIA	BPSQLSIA, START-UDS-PRINT-SQLSIA
BREORG	START-UDS-BREORG	BREORG, START-UDS-REORGANIZATION

BSTATUS	START-UDS-BSTATUS	BSTATUS
DDL	START-UDS-DDL	DDL
DMLTEST	START-UDS-DMLTEST	DMLTEST
SSL	START-UDS-SSL	SSL
ONLINE-PRIVACY	START-UDS-ONLINE-PRIVACY	OPRIVACY
UDSADM	START-UDS-ADM	UDSADM, START-UDS-ADMINISTRATION
UDSMON	START-UDS-UDSMON	UDSMON
UDS-Online-Utility	START-UDS-ONLINE-UTILITY	ONLUTIL

Tabelle 10: UDS/SQL - Programme uber START-Kommandos aufrufen

Syntax der START-UDS-...-Kommandos

START-UDS-...

VERSION = *STD / <product-version>

,MONJV = *NONE / <filename 1..54 without-gen-vers>

,CPU-LIMIT = *JOB-REST / <integer 1..32767 *seconds*>

,RESIDENT-PAGES = [***PAR**AMETERS](...) *Nur fur DBH*

[***PAR**AMETERS](...)

| **MIN**IMUM = ***STD** / <integer 0..32767 *4Kbyte*>

| ,**MAX**IMUM = ***STD** / <integer 0..32767 *4Kbyte*>

VERSION =
Produktversion des Programms, die gestartet werden soll.

VERSION = *STD
Keine explizite Angabe der Produktversion. Die Produktversion wird folgendermaßen ausgewählt:

1. Die mit dem Kommando /SELECT-PRODUCT-VERSION vorgegebene Version.
2. Die höchste mit IMON installierte Version.

VERSION = <product-version>
Explizite Angabe der Produktversion in der Form mm.n[a[kk]].

Es wird empfohlen, die Version immer vollständig anzugeben, z.B. 02.9B00, um den Versionswechsel bei Korrekturlieferungen zu erleichtern.

MONJV =
Angabe einer Monitor-Jobvariablen zur Überwachung des Programmlaufs.

MONJV = *NONE
Es wird keine Monitor-Jobvariable verwendet.

MONJV = <filename 1..54 without-gen-vers>
Name der zu verwendenden Jobvariablen.

Während des Programmlaufs setzt das System die Jobvariable auf folgende Werte:

$R	Programm läuft
$T	Programm erfolgreich beendet
$A	Programm fehlerhaft beendet

CPU-LIMIT =
Maximale CPU-Zeit in Sekunden, die das Programm beim Ablauf verbrauchen darf.

CPU-LIMIT = *JOB-REST
Es soll die verbleibende CPU-Zeit des BS2000-Jobs für die Aufgabe verwendet werden.

CPU-LIMIT = <integer 1..32767 *seconds* >
Es soll nur die angegebene Zeit verwendet werden.

RESIDENT-PAGES = *PARAMETERS(...)
Dieser Operand ist nur für den DBH erlaubt.

Anzahl residente Speicherseiten, die für den DBH-Lauf benötigt werden.
Der Operand muss angegeben werden, wenn im Programm mit einem CSTAT-Makro (siehe Handbuch „Makroaufrufe an den Ablaufteil") Seiten resident gemacht werden sollen. Die zulässige Anzahl an residenten Speicherseiten kann vom Operator beeinflusst werden. Fehlt der Operand (entspricht MIN=*STD,MAX=*STD), so werden die Speicheranforderungen dem Anfangssatz des Programms entnommen, wozu die Datei eröffnet werden muss.

MINIMUM = *STD / <integer 0..32767 *4Kbyte* >
Minimal benötigte Anzahl an residenten Speicherseiten.

MAXIMUM = *STD / <integer 0..32767 *4Kbyte* >
Maximal benötigte Anzahl an residenten Speicherseiten.

Nicht mit IMON installierte UDS/SQL-Programme

Wenn UDS/SQL nicht mit IMON installiert wurde, müssen Sie zum Starten der UDS/SQL-Programme folgende Kommandos angeben:

```
[/MODIFY-SDF-OPTIONS SYNTAX-FILE=$kennung.SYSSDF.UDS-SQL.029.USER]

 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname

 /ADD-FILE-LINK LINK-NAME=$UDSLIB,FILE-NAME=$kennung.SYSLNK.UDS-SQL.029

 /START-EXECUTABLE-PROGRAM FROM-FILE=(LIB=$kennung.SYSLNK.UDS-SQL.029
  ,ELEMENT=udsdienstprogramm)
```

Die Zuweisung der USER-Syntaxdatei mit dem Kommando MODIFY-SDF-OPTIONS wird nur benötigt, wenn die System-Syntax-Datei SYSSDF.UDS-SQL.029 nicht aktiv ist und wenn UDS/SQL-Programme mit SDF-Anweisungsoberfläche genutzt werden, also für

- BMEND
- BOUTLOAD
- BPGSIZE
- BPRIVACY und OPRIVACY
- BPSQLSIA
- BREORG
- UDSADM
- BTRANS24

2.4 Tools für UDS/SQL

Als Service-Leistung werden Tools mitausgeliefert, die über den UDS/SQL-Produktumfang hinausgehen. Die Beschreibung dieser Tools entnehmen Sie bitte den im Lieferumfang enthaltenen Informationsdateien.

Die Tools unterliegen keiner Wartungsverpflichtung und können von Fujitsu Technology Solutions ohne Vorankündigung geändert oder gestrichen werden.

3 Datenbank aufbauen (BCREATE, BFORMAT, DDL- und SSL-Compiler, BGSIA, BGSSIA, BCALLSI)

Eine Datenbank wird stufenweise aufgebaut. Nacheinander müssen Sie die im Bild 6 aufgeführten Vorbereitungen treffen und die angegebenen Programme ablaufen lassen.

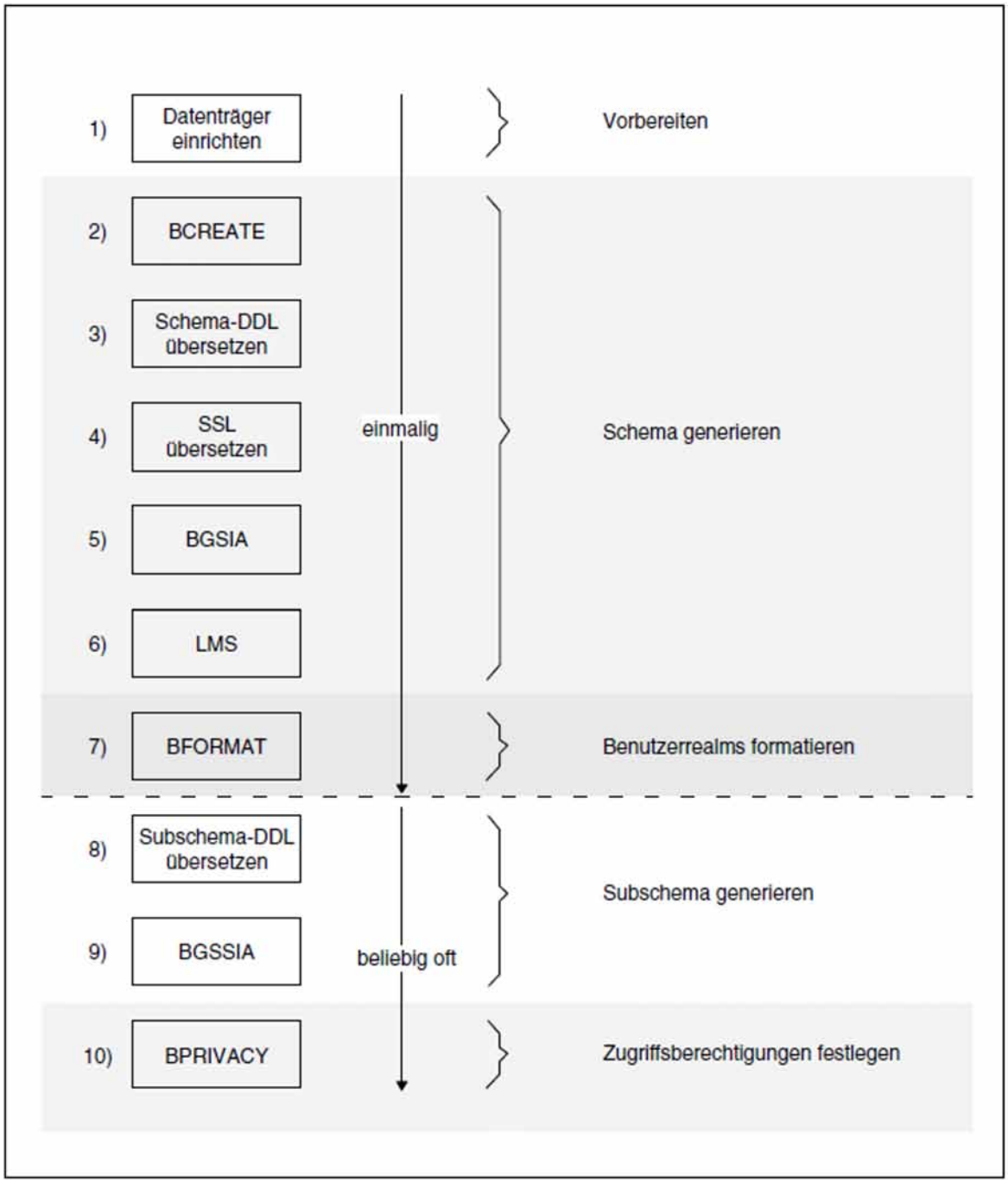

Bild 6: Übersicht zum Aufbauen einer Datenbank

Bild 7: Zusatzmaßnahmen beim Aufbauen einer Datenbank

Vorbereiten

1. Mit dem CREATE-FILE-Kommando legen Sie den Speicherplatz fest für die Realms und Dateien der Compilerdatenbank: DBDIR, DBCOM, COSSD und DBSTAT.

Schema generieren

2. Das Dienstprogramm BCREATE formatiert die Realms der Compilerdatenbank: DBDIR und DBCOM.
3. Der DDL-Compiler übersetzt Ihre Schema-DDL und speichert das Ergebnis der Übersetzung im DBCOM. Außerdem beschreibt der DDL-Compiler die bereits eingerichtete Datei COSSD.
4. Die Übersetzung der SSL erfolgt wahlweise, je nachdem, ob Sie die Standardwerte für die Speicherstruktur verwenden oder mit Hilfe der SSL die physische Struktur der Datenbank beeinflussen wollen. Wenn Sie eine Speicherbeschreibung (SSL) erstellt haben, müssen Sie an dieser Stelle die SSL übersetzen.
   Der SSL-Compiler legt die Satz- und Seteinträge im DBCOM entsprechend neu fest.
5. Das Dienstprogramm BGSIA baut die Schema Information Area (SIA) auf mit Hilfe der Einträge aus dem DBCOM und speichert sie in das DBDIR. Außerdem erzeugt BGSIA das Modul UDSHASH und legt es in der EAM-Datei ab.
6. Das durch BGSIA erzeugte Modul UDSHASH müssen Sie durch das BS2000-Dienstprogramm LMS in die Modulbibliothek *dbname*.HASHLIB eintragen (siehe Handbuch „LMS (BS2000)“).

Benutzerrealms formatieren

7. Das Dienstprogramm BFORMAT formatiert die Benutzerrealms der Datenbank mit den Informationen, die im DBDIR gespeichert sind. Anschließend stehen die Daten in der Schema Information Area (SIA).

Subschema generieren

8. Der DDL-Compiler übersetzt die Subschema-DDL und legt sie im DBCOM und COSSD ab.
9. Das Dienstprogramm BGSSIA baut die Subschema Information Area (SSIA) auf und speichert sie im DBDIR.

Zugriffsberechtigung festlegen

10. Die Zugriffsberechtigungen legen Sie mit einem der Dienstprogramme ONLINE-PRIVACY oder BPRIVACY fest (siehe Kapitel „Zugriffsberechtigungen festlegen (ONLINE-PRIVACY, BPRIVACY)").

SSITAB-Modul erzeugen

Das Dienstprogramm BCALLSI benötigen nur Anwender der CALL-DML. BCALLSI erzeugt das SSITAB-Modul mit den Subschema-Informationen, die das CALL-DML-Anwenderprogramm braucht.

3.1 Datenbank-Aufbau vorbereiten

Die Vorbereitungen zum Aufbauen der Datenbank gliedern sich in:

- Compilerdatenbank einrichten
- Benutzerdatenbank einrichten

Wenn Sie den Ort der Speicherung auf gemeinschaftlicher Platte durch den Einsatz von MPVS bestimmen wollen, beachten Sie bitte die Hinweise im Handbuch „Datenbankbetrieb“, MPVS für UDS/SQL nutzen.

3.1.1 Compilerdatenbank einrichten

Die Compilerdatenbank (siehe Abschnitt „Dateien und Realms einer UDS/SQL-Datenbank") besteht aus folgenden Realms:

DBDIR Database Directory

DBCOM Database Compiler Realm

aus der Datei:

COSSD COBOL Subschema Directory

Mit dem BS2000-Kommando CREATE-FILE müssen Sie diese Dateien und Realms einrichten und ihre Größe festlegen.

DBDIR und DBCOM einrichten

```
/CREATE-FILE FILE-NAME=dbname.DBDIR
     ,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primär
     ,SECONDARY-ALLOCATION=sekundär))
    [,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,DEVICE-TYPE=gerät,SPACE=...)]

/CREATE-FILE FILE-NAME=dbname.DBCOM
     ,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primär
     ,SECONDARY-ALLOCATION=sekundär))
    [,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,DEVICE-TYPE=gerät,SPACE=...)]

/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
```

dbname

Name der Datenbank mit max. 17 Zeichen Länge. *dbname* kommt als Teilqualifikation in fast allen Dateinamen der Datenbankdateien vor und muss folgenden Regeln entsprechen:

- *dbname* muss bei allen Dateien der Datenbank gleich sein.
- *dbname* darf weder Sonderzeichen noch Leerzeichen enthalten und das erste Zeichen muss ein Buchstabe sein.
- :*catid*:$*userid*.*dbname*.*realmname*.*copyname* darf insgesamt maximal 54 Zeichen lang sein.

Bei UDS-D gilt:

- *dbname* muss netzweit eindeutig sein.

SPACE

Festlegen der Speicherplatzgröße (siehe „Maximalgröße für UDS/SQL-Dateien“, „Dateien und Realms einer UDS/SQL-Datenbank").

PRIMARY-ALLOCATION=*primär*

Primärzuweisung:
Die Formatierung von DBCOM und DBDIR mit dem Dienstprogramm BCREATE (siehe Abschnitt „Compilerdatenbank formatieren mit BCREATE") setzt folgende Mindestwerte für die Primärzuweisung voraus:

	DBDIR	DBCOM
2-Kbyte-Format	52	100
4-Kbyte-Format	79	424
8-Kbyte-Format	127	607

Für Datenbanken mit umfangreichem Schema ist entsprechend mehr Platz zuzuweisen oder durch Sekundärzuweisung > 0 die automatische Erweiterbarkeit zu ermöglichen. Später kann mit dem Dienstprogramm BREORG der ungenutzte Speicherplatz reduziert werden (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BREORG).

SECONDARY-ALLOCATION=*sekundär*

Sekundärzuweisung;

sekundär=0

Mit dieser Einstellung wird die Möglichkeit der automatischen Realm-Erweiterung bzw. der Online-Realm-Erweiterung unterdrückt.

sekundär>0

ist Voraussetzung für automatische Realm-Erweiterbarkeit bzw. Online-Realm-Erweiterbarkeit und Voraussetzung für die Aktivierung der Online-Erweiterbarkeit des DBDIR mit dem Kommando ACT INCR.
Im Falle sekundär>0 wird für den betreffenden Realm bereits beim DB-Aufbau die Online-Realm-Erweiterbarkeit aktiviert.
Bei einer so zustande gekommenen Aktivierung der Online-Realm-Erweiterbarkeit werden für NR-PAGES und MIN-PAGES die Standardwerte NR-PAGES=64 und MIN-PAGES=0 eingetragen (siehe Aktivierung der Online-Erweiterbarkeit beim Datenbankaufbau und DAL-Kommando ACT INCR im Handbuch „Datenbankbetrieb“).

VOLUME
DEVICE-TYPE

Wenn Sie DBDIR und DBCOM auf private Platte (PRIVATE VOLUME) legen, müssen Sie angeben:

priv-vsn

Archivnummer des Datenträgers

gerät

Gerätetyp der privaten Platte

LINK-NAME

Das Database Directory müssen Sie über den Dateikettungsnamen DATABASE mit der Datenbank verbinden.

COSSD einrichten

```
/CREATE-FILE FILE-NAME=dbname.COSSD
    ,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primär
    ,SECONDARY-ALLOCATION=sekundär))
   [,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,DEVICE-TYPE=gerät,SPACE=...)]
```

SPACE

Festlegen der Speicherplatzgröße (siehe „Maximalgröße für UDS/SQL-Dateien“, „Dateien und Realms einer UDS/SQL-Datenbank").

PRIMARY-ALLOCATION=*primär*

Primärzuweisung;
der Speicherplatzbedarf des COSSD hängt von Anzahl und Größe der übersetzten Subschemata ab.
Um später ggf. weitere übersetzte Subschemata ins COSSD aufnehmen zu können, empfiehlt es sich, das COSSD mit einer Primärzuweisung und Sekundärzuweisung (s.u.) von jeweils 100 2K-Einheiten (BS2000-Halfpages) anzulegen.

SECONDARY-ALLOCATION=*sekundär*

Sekundärzuweisung;
abhängig von der Generierung des Betriebssystems ist auch COSSD nicht unbegrenzt dynamisch erweiterbar.
Es empfiehlt sich, 100 2K-Einheiten (BS2000-Halfpages) zuzuweisen.

VOLUME
DEVICE-TYPE

Wenn Sie COSSD auf private Platte (PRIVATE VOLUME) legen:

priv-vsn

Archivnummer des Datenträgers

gerät

Gerätetyp der privaten Platte

3.1.2 Benutzerrealms einrichten

Die Benutzerrealms müssen Sie, ebenso wie die Bereiche der Compilerdatenbank, mit dem CREATE-FILE-Kommando einrichten.

Wenn Sie vor dem Übersetzen der Schema-DDL und SSL noch nicht die Größe der Benutzerrealms abschätzen können, genügt es, nach dem Dienstprogramm BGSIA und vor dem Dienstprogramm BFORMAT die Benutzerrealms einzurichten.
BGSIA druckt den ESTIMATE-REPORT aus, in dem von UDS/SQL geschätzte Größen der einzelnen Benutzerrealms stehen (siehe Abschnitt „Beschreibung des ESTIMATE-REPORT").

```
/CREATE-FILE FILE-NAME=dbname.realmname
     ,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primär
     ,SECONDARY-ALLOCATION=sekundär))
    [,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,DEVICE-TYPE=gerät,SPACE=...)]
```

dbname

Datenbankname

realmname

Name des Benutzerrealm, der in der Schema-DDL definiert wurde

SPACE

Festlegen der Speicherplatzgröße (siehe „Maximalgröße für UDS/SQL-Dateien“, „Dateien und Realms einer UDS/SQL-Datenbank").

PRIMARY-ALLOCATION=*primär*

Primärzuweisung für den Benutzerrealm.

SECONDARY-ALLOCATION=*sekundär*

Sekundärzuweisung.

sekundär=0

unterdrückt die automatische Realm-Erweiterung bzw. die Online-Realm-Erweiterung.

sekundär>0

ist Voraussetzung für automatische Realm-Erweiterbarkeit bzw. Online-Realm-Erweiterbarkeit.
Im Falle sekundär>0 wird für den betreffenden Realm bereits beim DB-Aufbau die Online-Realm-Erweiterbarkeit aktiviert.
Bei einer so zustande gekommenen Aktivierung der Online-Realm-Erweiterbarkeit werden für NR-PAGES und MIN-PAGES die Standardwerte NR-PAGES=64 und MIN-PAGES=0 eingetragen (siehe Aktivierung der Online-Erweiterbarkeit beim Datenbankaufbau und DAL-Kommando ACT INCR im Handbuch „Datenbankbetrieb“).

VOLUME
DEVICE-TYPE

Wenn Sie den Benutzerrealm auf private Platte (PRIVATE VOLUME) legen:

priv-vsn

Archivnummer des Datenträgers

gerät

Gerätetyp der privaten Platte

3.2 Schema generieren

Um das Schema zu generieren, müssen Sie nacheinander folgende Programme ablaufen lassen:

BCREATE	Compilerdatenbank formatieren
DDL-Compiler	Schema-DDL übersetzen
SSL-Compiler	SSL übersetzen
BGSIA	Schema Information Area (SIA) einrichten

3.2.1 Compilerdatenbank formatieren mit BCREATE

Mit dem Dienstprogramm BCREATE werden die Realms DBDIR und DBCOM der Compilerdatenbank formatiert. Durch BCREATE erhalten DBDIR und DBCOM eine Act-Key-0-Seite (Sicherungsinformation, Erstellungsdatum etc.) und mindestens eine FPA-Seite (Freiplatzverwaltung).

BCREATE erweitert bei Bedarf automatisch die Realms der bearbeiteten Datenbank. Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme).

BCREATE berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

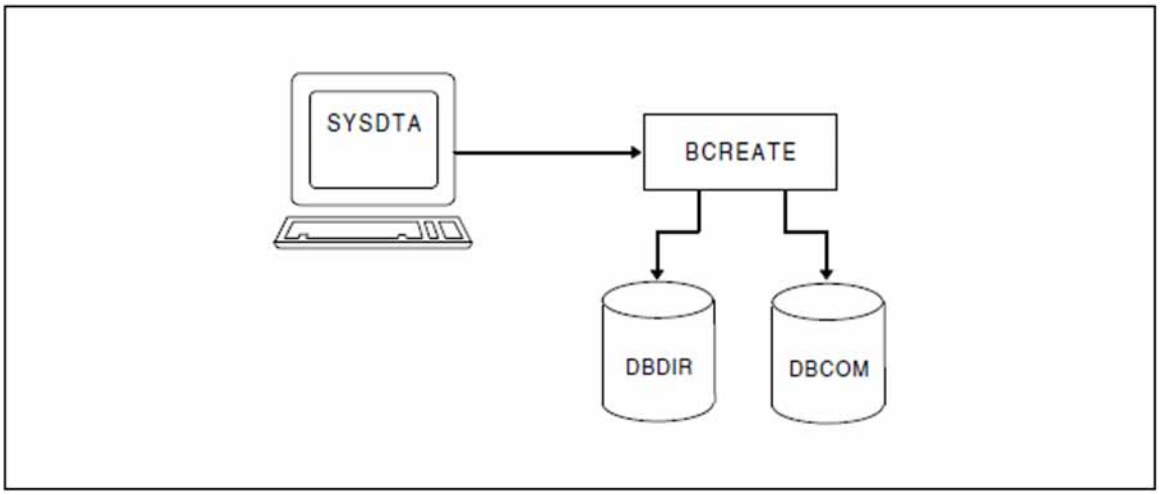

Bild 8: Systemumgebung von BCREATE

Eine UDS/SQL-Datenbank können Sie wahlweise mit 2-Kbyte-, 4-Kbyte- oder 8-Kbyte-Seitenformat aufbauen. Im 4-Kbyte-Seitenformat und im 8-Kbyte-Seitenformat ist jede Datenbankseite in einen Seitencontainer eingebettet (siehe Handbuch „Entwerfen und Definieren"). Für die Länge der Datenbankseiten ergeben sich folgende Werte:

- 2048 Byte bei Datenbanken mit 2-Kbyte-Seitenformat
- 4000 Byte bei Datenbanken mit 4-Kbyte-Seitenformat (Seitencontainer: 4096 byte)
- 8096 Byte bei Datenbanken mit 8-Kbyte-Seitenformat (Seitencontainer: 8192 byte)

Anweisungen für BCREATE

Anweisung	Standardwert	Bedeutung
`DATABASE-PAGE-LENGTH IS { 2KB | 4KB | 8KB }`	4KB	wahlweise; legt die Seitenlänge für die neue Datenbank fest: • 2KB Die Datenbank wird mit 2-Kbyte-Seitenformat aufgebaut • 4KB Die Datenbank wird mit 4-Kbyte-Seitenformat aufgebaut • 8KB Die Datenbank wird mit 8-Kbyte-Seitenformat aufgebaut
`END`	-	generell erforderlich; beendet die Eingabe der Anweisungen

Tabelle 11: Anweisungen für BCREATE

Kommandofolge zum Starten von BCREATE

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 /CREATE-FILE FILE-NAME=dbname.DBDIR ...
02 /CREATE-FILE FILE-NAME=dbname.DBCOM ...
03 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
04 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
05 /START-UDS-BCREATE
06 [DATABASE-PAGE-LENGTH IS {2/4/8}KB]
07 END
```

01/02 siehe „DBDIR und DBCOM einrichten“, "Compilerdatenbank einrichten".

04 Die angegebene Version von BCREATE wird ausgewählt. Die Angabe der Version wird generell empfohlen, da mehrere UDS/SQL-Versionen parallel installiert sein können.

05 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen BCREATE gestartet werden.

06 Nur wenn die Datenbank mit 4-Kbyte-Seitenformat aufgebaut werden soll, kann die DATABASE-PAGE-LENGTH-Anweisung entfallen.

07 Die END-Anweisung ist erforderlich.

Beispiel

```
/CREATE-FILE FILE-NAME=REISEN.DBDIR,SUPPORT=PUBLIC-DISK(SPACE= -
/     RELATIVE(PRIMARY-ALLOCATION=150,SECONDARY-ALLOCATION=50))
/CREATE-FILE FILE-NAME=REISEN.DBCOM,SUPPORT=PUBLIC-DISK(SPACE=RELATIVE -
/     (PRIMARY-ALLOCATION=500,SECONDARY-ALLOCATION=50))
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=REISEN.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BCREATE
***** START        BCREATE      (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:01
+++++ WARNING: 1917 BLOCKLENGTH SET TO 4KB
* SCHEMAS AND SUBSCHEMAS WRITTEN TO DBDIR
* VERSION-RECORDS WRITTEN TO DBDIR
* DBCOM SUCCESSFULLY FORMATTED
* DBDIR SUCCESSFULLY FORMATTED

***** DIAGNOSTIC SUMMARY OF BCREATE

+++++           1 WARNINGS
               NO ERRORS
               NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          69
***** NORMAL END   BCREATE      (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:01
```

3.2.2 Schema-DDL übersetzen

Die Schema-DDL übersetzen Sie mit dem DDL-Compiler; Sie müssen die Schema-DDL dem DDL-Compiler als Eingabedatei zuweisen.
Nach erfolgreichem Übersetzungslauf speichert der DDL-Compiler die Schema-Beschreibung im DBCOM (Database Compiler Realm).

Mit Hilfe dieser Information baut das nachfolgende Dienstprogramm BGSIA die SIA auf und speichert sie im DBDIR (Database Directory).

Wenn Sie eine SSL-Beschreibung erstellt haben, ist die übersetzte Schema-Beschreibung im DBCOM die Grundlage für die Weiterverarbeitung durch den SSL-Compiler und das Dienstprogramm BGSIA.
Haben Sie keine SSL-Beschreibung erstellt, dann ist die Schema-Beschreibung im DB-COM abgeschlossen.

Der DDL-Compiler erstellt auch das COBOL Subschema Directory (COSSD). Das COSSD speichert Informationen für den COBOL-Compiler, die beim Generieren der DB-Anwenderprogramme erforderlich sind. Erst beim Übersetzungslauf der Subschema-DDL wird der eigentliche Inhalt des COSSD generiert.

Der DDL-Compiler erweitert bei Bedarf automatisch DBDIR und DBCOM der bearbeiteten Datenbank bzw. die DBTTs der Satzarten in DBDIR und DBCOM. Näheres hierzu siehe Handbuch „Datenbankbetrieb“, Automatische Realm-Erweiterung durch Dienstprogramme).

Der DDL-Compiler berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb“, Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

Während des Ablaufs benutzt der DDL-Compiler den linked-in DBH.

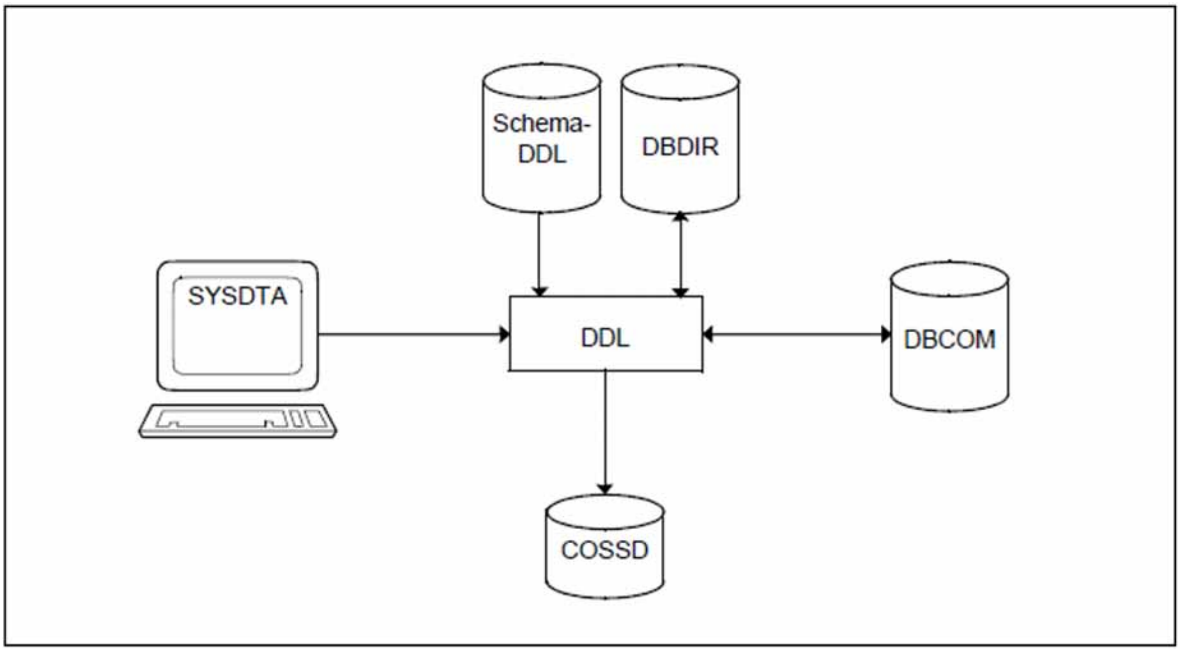

Bild 9: Systemumgebung bei der Schema-DDL-Übersetzung

Anweisungen für die Übersetzung

Mit dem DDL-Compiler übersetzen Sie nicht nur die Schema-DDL, sondern auch die Subschema-DDL; die SSL dagegen übersetzt der SSL-Compiler.
In der folgenden Tabelle finden Sie die Anweisungen für die Übersetzung der

- Schema-DDL gekennzeichnet mit DDL
- Subschema-DDL gekennzeichnet mit SDDL
- SSL gekennzeichnet mit SSL

Anweisung	Compiler	Standard-wert	Bedeutung
`PARLIST IS { YES | NO }`	DDL SDDL SSL	NO	wahlweise; YES alle Anweisungen werden auf SYSLST aufgelistet NO die Anweisungen werden nicht aufgelistet
`SORCLIST IS { YES | NO }`	DDL SDDL SSL	YES	wahlweise; YES auf SYSLST wird ein Protokoll ausgedruckt, u.U. mit Fehlermeldungen NO kein Protokoll wird ausgedruckt

`SOURCE IS`
`{ '`*`dateiname`*`' |`
`'`*`bib`*`(`*`element`*`)' }`
	DDL
	SDDL
	SSL
		-	nicht erforderlich, wenn die Eingabe im Dialog erfolgt oder die Eingabedatei SYSDTA zugewiesen ist - in diesem Fall ist zu beachten, dass zunächst alle Anweisungen (mindestens END) und anschließend die eigentliche Source eingegeben wird.
			
			weist dem Compiler die Datei zu, die die Schema-DDL/Subschema-DDL/SSL enthält. An Stelle von '*dateiname*' kann auch ein Element einer Programmbibliothek angegeben werden (siehe Handbuch „LMS (BS2000)“, Programmbibliotheken).
			
			bib: Name der Programmbibliothek
			element: Name des Elements
			
			SYSDTA wird auf die Eingabedatei umgeschaltet. Nach Beendigung des Compiler-Laufs wird SYSDTA wieder auf SYSCMD zurückgesetzt. Die Anweisungen „SOURCE IS“ und „DELETE SCHEMA“ bzw. „DELETE SUBSCHEMA“ dürfen Sie nicht innerhalb desselben DDL-Compiler-Laufs verwenden.

SUBSCHEMA FORM IS OLD	SDDL	-	wahlweise; diese Anweisung wird nur noch für Subschemata benötigt, die von KDBS-Anwendungen genutzt werden; sie ist nur zulässig in Verbindung mit der Anweisung „SOURCE IS *dateiname*“ und wird bei der Schemaübersetzung ignoriert. Die Anweisung „SUBSCHEMA FORM IS OLD“ bewirkt, dass das transformierte Subschema und die Prüftabelle (CHECK-TABLE) in einem internen Format in das COSSD eingetragen wird, das bis einschließlich UDS/SQL V1.2 das Standardformat war („alte“ Form; alle Referenznummern sind 1 byte lang). Ein Subschema kann nur dann in eine Form gemäß UDS/SQL V1.2 übersetzt werden, wenn die folgenden Bedingungen erfüllt sind: • Kein Feld des Subschemas hat den Typ DATABASE-KEY-LONG. • Kein Feld des Subschemas hat den Typ NATIONAL. • Keine Satzart des Subschemas ist länger als 2020 byte. • Alle Satzart- und Setnummern des Schemas sind <= 254. Andernfalls beendet sich der DDL-Compiler mit Syntaxfehler und das Subschema wird nicht in DBCOM und COSSD eingetragen.

GENERATE-REC-REF IS { YES | NO }	SDDL	NO	wahlweise; YES Satzreferenzen werden erzeugt: In der Struktur IMPLICITLY-DEFINED-DATA-NAMES wird ein Feld REC-REF PIC S9(4) BINARY definiert. Für jede Satzreferenz wird diesem Feld ein Bedingungsname (Stufennummer 88) zugeordnet, der nach folgendem Muster aufgebaut ist: REF-*Satzname*. Da die Maximallänge für einen Namen 30 Zeichen beträgt, wird *Satzname*, falls erforderlich, nach dem 26. Zeichen abgeschnitten. In diesem Fall muss *Satzname* in den ersten 26 Zeichen eindeutig sein. Die Satzreferenz kann in einem COBOL-Programm wie folgt verwendet werden: SET REF-*Satzname* IN REC-REF TO TRUE. MOVE REC-REF TO *dbkey*. NO Es werden keine Satzreferenzen erzeugt Diese Anweisung ist nur für das Generieren von Subschemas wirksam. Beim Generieren von Schemas wird die Anweisung ignoriert.
DELETE SCHEMA '*schemaname*'	DDL	-	wahlweise; das angegebene Schema wird wieder gelöscht; sinnvoll, wenn nach der Umstrukturierung mit BALTER die DDL zwar richtig läuft, aber die SSL-Übersetzung Fehler anzeigt, die eigentlich auf die DDL zurückzuführen sind *schemaname*: Name des Schemas Die Anweisungen „SOURCE IS“ und „DELETE SCHEMA“ dürfen Sie nicht innerhalb desselben DDL-Compiler-Laufs verwenden.

DELETE [ONLY] SUBSCHEMA '*subschemaname*' {OF | :} SCHEMA '*schemaname*'	SDDL	-	wahlweise; das angegebene Subschema wird gelöscht. Das Subschema, das übersetzt wird, und das Subschema, das Sie durch die DELETE-Anweisung benennen, kann denselben Namen haben, da es vor dem Übersetzungslauf gelöscht wird. ONLY wird der Parameter weggelassen, muss nach der DELETE-Anweisung eine SOURCE-Anweisung folgen. Ist der Parameter angegeben, wird eine SOURCE-Anweisung ignoriert. *subschemaname*: Name des Subschemas *schemaname*: Name des Schemas beide Namen müssen in Hochkommata eingeschlossen werden
DISPLAY IS { YES | NO }	DDL SDDL SSL	NO	wahlweise; YES die verschiedenen Informationen über Satzarten, Sets usw. aus dem DBCOM werden in Klartext ausgegeben. NO Werte aus dem DBCOM werden nicht ausgegeben
CREATE COSSD '*schemaname*'	DDL SDDL	-	nachträglich das COSSD einrichten; wenn dies bei der Schemaübersetzung vergessen wurde oder der DDL-Compiler wegen eines Fehlers beim Einrichten des COSSD abnormal beendet wurde, so kann es bis zur Subschema-Übersetzung in einem eigenen Lauf nachgeholt werden; *schemaname* muss in Hochkommata eingeschlossen werden. Das COSSD muss vor dem Compilerlauf per CREATE-FILE-Kommando eingerichtet werden. Hinweis: Wenn Sie gleichzeitig den Parameter SOURCE IS ... angeben, findet keine Übersetzung statt.

COMPARE SUBSCHEMAS	SDDL	-	gilt nur nach einer Umstrukturierung mit BALTER; die Subschemata des alten Schemas werden auf Verträglichkeit zum neuen Schema geprüft; dazu liest der DDL-Compiler die Subschemata aus dem alten COSSD nach einem BALTER-Lauf heraus. Ist das alte Subschema kompatibel zum neuen Schema, wird das Subschema in den neuen DBCOM und das neue COSSD eingetragen.
DIAGNOSTIC IS { YES | NO }	SDDL	NO	nur sinnvoll zusammen mit COMPARE; YES Unverträglichkeiten der zum neuen Schema inkompatiblen Subschemata werden diagnostiziert und in Form von Fehlermeldungen aufgelistet NO es werden keine Fehlermeldungen ausgegeben
QUOTE IS { SINGLE | DOUBLE }	DDL SDDL	DOUBLE	wahlweise; SINGLE Literale in der Schema-DDL/Subschema-DDL werden in Apostrophen eingeschlossen DOUBLE Literale in der Schema-DDL/Subschema-DDL werden in Anführungszeichen eingeschlossen
END	DDL SDDL SSL	-	generell erforderlich; schließt die Eingabe der Anweisungen ab

Tabelle 12: Anweisungen zum Ubersetzen der Schema-DDL/Subschema-DDL/SSL

Kommandofolge zum Übersetzen der Schema-DDL

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 /CREATE-FILE FILE-NAME=dbname.COSSD ...
02 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
03 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
04 /CREATE-FILE FILE-NAME=dbname.DBSTAT,SUPPRESS-ERRORS=*FILE-EXISTING
   /CREATE-FILE FILE-NAME=dbname.DBSTAT.SAVE,SUPPRESS-ERRORS=*FILE-EXISTING
05 /START-UDS-DDL
06 ddl-compiler-anweisungen
07 END
```

01 siehe „COSSD einrichten“ ("Compilerdatenbank einrichten")

03 Versionsabhängiges Modul des linked-in DBH der entsprechenden Version wird nachgeladen (siehe Handbuch „Anwendungen programmieren“, Abschnitt „UDS/SQL-TIAM-Anwenderprogramme binden, laden und starten“).

04 Die DB-Status-Dateien benötigt der DBH. Falls der Datenbankname mehr als 8 Zeichen enthält, dürfen in diesen CREATE-FILE-Anweisungen für *dbname* nur die ersten 8 Zeichen des Datenbanknamens angegeben werden.

05 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen DDL gestartet werden.

06 Die einzelnen Anweisungen können, durch Kommas oder Leerzeichen getrennt, in einer Zeile eingegeben werden.

Beispiel

```
/CREATE-FILE FILE-NAME=REISEN.COSSD,SUPPORT=PUBLIC-DISK(SPACE=RELATIVE -
/   (PRIMARY-ALLOCATION=30,SECONDARY-ALLOCATION=10))
/CREATE-FILE FILE-NAME=REISEN.DBSTAT,SUPPORT=PUBLIC-DISK(SPACE=RELATIVE -
/   (PRIMARY-ALLOCATION=24,SECONDARY-ALLOCATION=48))
/CREATE-FILE FILE-NAME=REISEN.DBSTAT.SAVE,SUPPORT=PUBLIC-DISK(SPACE=RELATIVE -
/   (PRIMARY-ALLOCATION=24,SECONDARY-ALLOCATION=48))
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=REISEN.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-DDL
 ***** START       DDLCOMP        (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:06
 *  DDLCOMP: INPUT SYSTEMPARAMETERS
 SOURCE IS 'S.REISEN.DDL'
 END
 *  DDLCOMP: READ SCHEMA/SUBSCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:26:06/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:26:06/0YA2)
 0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.PUBS
 0YA2: PUBSETS:        SQL2
 0YA2: DEFAULT PUBSET: SQL2
 0YA2: ----------------------------------------
 *  DDLCOMP: START SCHEMA-PHASE
 *  DDLCOMP: CHECK SCHEMA RULES
 *  DDLCOMP: CHECK DATA ALLOCATION
 *  DDLCOMP: SEMANTIC TEST
 *  DDLCOMP: CYCLUS TESTS
 *  DDLCOMP: ERROR DIAGNOSTIC
 *  DDLCOMP: NO ERRORS IN SCHEMA-PHASE
 *  DDLCOMP: CREATE FILE COSSD
 *  DDLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:26:06/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: REISEN                 1394       4046         67       1760         45
 %  UDS0213 UDS NORMAL BEENDET MIT ************1394 DML-STATEMENTS 2017-06-28 (ILLY033,11:26:
06/0YA2)

 ***** DIAGNOSTIC SUMMARY FOR DDL-SCHEMA REISEBUERO

               NO ERRORS
 +++++          8 WARNINGS

 ***** END OF DIAGNOSTIC SUMMARY
 ***** NORMAL END   DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:06
```

3.2.3 SSL übersetzen

Das Übersetzen der Speicherstruktur-Beschreibung ist wahlfrei; ohne SSL nimmt UDS/SQL Standardwerte an. Wenn Sie eine SSL-Beschreibung erstellt haben, können Sie diese SSL durch den SSL-Compiler übersetzen lassen.

Der SSL-Compiler analysiert die Speicherstruktur-Beschreibung und ändert, entsprechend der Angaben in der SSL, die Einträge im DBCOM.

Der SSL-Compiler erweitert bei Bedarf automatisch DBDIR und DBCOM der bearbeiteten Datenbank bzw. die DBTTs der Satzarten in DBDIR und DBCOM. Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme).

Der SSL-Compiler berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

Während des Ablaufs benutzt der SSL-Compiler den linked-in DBH.

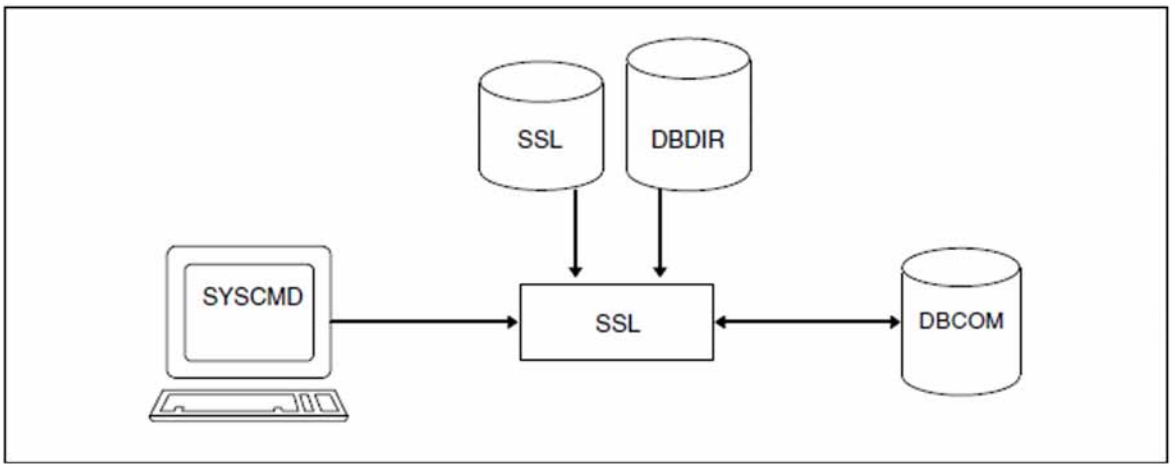

Bild 10: Systemumgebung bei der SSL-Übersetzung

Anweisungen des SSL-Compilers

Die Anweisungen des SSL-Compilers sind in der Tabelle der Compiler-Anweisungen (siehe Tabelle 12 im "Schema-DDL übersetzen") enthalten.

Kommandofolge zum Übersetzen der SSL

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /START-UDS-SSL
04 ssl-compiler-anweisungen
05 END
```

02 Versionsabhängiges Modul des linked-in DBH der entsprechenden Version wird nachgeladen (siehe Handbuch „Anwendungen programmieren", Abschnitt „UDS/SQL-TIAM-Anwenderprogramme binden, laden und starten“).

03 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen SSL gestartet werden.

04 Die einzelnen Anweisungen können, durch Kommas oder Leerzeichen getrennt, in einer Zeile eingegeben werden.

Beispiel

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=REISEN.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-SSL
***** START       SSLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:06
 *  SSLCOMP: INPUT SYSTEMPARAMETERS
 SORCLIST IS YES
 SOURCE IS 'S.REISEN.SSL'
 END
 *  SSLCOMP: READ SSL-SCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:26:06/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:26:06/0YA2)
 0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.PUBS
 0YA2: PUBSETS:       SQL2
 0YA2: DEFAULT PUBSET: SQL2
 0YA2: ----------------------------------------
 *  SSLCOMP: START SSL-PHASE
 *  SSLCOMP: CHECK SSL RULES
 *  SSLCOMP: SEMANTIC TEST
 *  SSLCOMP: ERROR DIAGNOSTIC
 *  SSLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:26:06/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: REISEN                  354        563         62         57         27
%  UDS0213 UDS NORMAL BEENDET MIT **************354 DML-STATEMENTS 2017-06-28
(ILLY033,11:26:06/0YA2)

 ***** DIAGNOSTIC SUMMARY FOR SSL - SCHEMA

              NO ERRORS
              NO WARNINGS

 ***** END OF DIAGNOSTIC SUMMARY
 ***** NORMAL END   SSLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:06
```

3.2.4 Schema Information Area (SIA) erzeugen mit BGSIA

Die Schema Information Area (SIA) müssen Sie mit dem Dienstprogramm BGSIA im DBDIR (Database Directory) einrichten.

BGSIA benötigt dazu die Informationen, die während der Übersetzung der Schema-DDL und der SSL im DBCOM (Database Compiler Realm) abgelegt wurden. Die SIA enthält anschließend Informationen über das Datenbankschema und seine Speicherstruktur in Tabellenform.

Der DBH und andere Dienstprogramme benötigen die SIA, wenn Daten der Benutzerrealms gespeichert, wiedergewonnen oder modifiziert werden.

BGSIA weist den Namen der Realms, der Satzarten und der Sets sowie den Schlüsseln Referenznummern zu. Diese Nummern druckt BGSIA, wenn Sie es mit der DISPLAY-Anweisung anfordern, am Ende des Laufs aus. Dieses Protokoll entspricht dem des Dienstprogramms BPSIA (siehe Handbuch „Sichern, Informieren und Reorganisieren",
SIA PRINT REPORT).

BGSIA erzeugt außerdem das Modul UDSHASH und legt es in der EAM-Datei ab. Es enthält eine Tabelle mit den Namen aller Hashroutinen, die Sie in der Schema-DDL definiert haben. Sie müssen das Modul UDSHASH nach dem BGSIA-Lauf mit den Attributen RMODE=ANY und AMODE=ANY in eine Modulbibliothek mit dem Namen *dbname*.HASHLIB übernehmen; dies gilt auch, wenn Sie keine Hashroutine verwenden.

Haben Sie eigene Hashroutinen programmiert (siehe Handbuch „Entwerfen und Definieren", Direkter Zugriff), so müssen Sie diese Module ebenfalls in die HASHLIB eintragen.

BGSIA erweitert bei Bedarf automatisch den DBDIR der bearbeiteten Datenbank. Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme).

BGSIA berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

Während des Laufs benutzt das Dienstprogramm BGSIA den linked-in DBH.

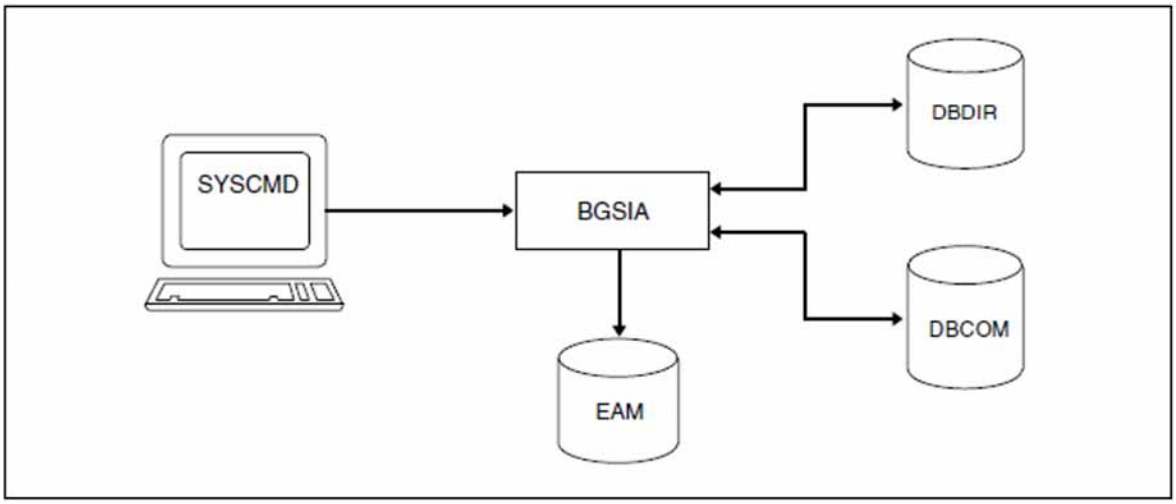

Bild 11: Systemumgebung von BGSIA

Anweisungen für BGSIA

Anweisung	Standardwert	Bedeutung
GENERATE SCHEMA *schemaname*	-	muss angegeben werden; prüfen und erstellen der SIA *schemaname*: Name des Schemas, der in der Schema-DDL angegeben wurde
RENAME { AREA | RECORD | SET } {'*name-old*' TO '*name-new*'} [, ...] .		darf nur im Umbenennungszyklus angegeben werden; Ändert die Namen von Satzarten, Sets und Benutzerrealms *name-old*: Name der geändert werden soll name-new neuer Name Umbenennungen und Änderungen von Feldern in Satzarten können hier nicht angegeben werden.
DISPLAY [SCHEMA *schemaname*]	-	wahlweise: ausdrucken der von BGSIA erzeugten SIA *schemaname*: Name des Schemas, der in der GENERATE-Anweisung angegeben wurde Die Angabe DISPLAY genügt.
END	-	generell erforderlich; schließt die Eingabe der Anweisungen ab

Tabelle 13: Anweisungen für BGSIA

Kommandofolge zum Starten von BGSIA

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 /DELETE-SYSTEM-FILE FILE-NAME=*OMF
02 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
03 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
04 /START-UDS-BGSIA
05 bgsia-anweisungen
06 END
```

03 Versionsabhängiges Modul des linked-in DBH der entsprechenden Version wird nachgeladen (siehe Handbuch „Anwendungen programmieren", Abschnitt „UDS/SQL-TIAM-Anwenderprogramme binden, laden und starten“).

04 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen BGSIA gestartet werden.

Modul UDSHASH in die HASHLIB eintragen

```
01 /START-LMS
02 //OPEN-LIB LIB=dbname.HASHLIB,MODE=*UPDATE(STATE=*NEW)
03 //ADD-ELEMENT FROM-FILE=*OMF,TO-ELEMENT=*LIBRARY-ELEMENT(TYPE=R)
04 //END
```

Beispiel

```
/DELETE-SYSTEM-FILE FILE-NAME=*OMF
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=REISEN.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BGSIA
***** START       BGSIA       (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:06
 GENERATE REISEBUERO
 DISPLAY
 END
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:26:07/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:26:07/0YA2)
 0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.PUBS
 0YA2: PUBSETS:       SQL2
 0YA2: DEFAULT PUBSET: SQL2
 0YA2: ----------------------------------------
 ESTIMATE-REPORT
********** FOR USER-REALM     3 NAME IS : REISENRLM
      A SIZE OF       147 BLOCKS WAS ESTIMATED
 END OF ESTIMATE-REPORT
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:26:07/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: REISEN                 1179       1310         61        276         35
%  UDS0213 UDS NORMAL BEENDET MIT ************1179 DML-STATEMENTS 2017-06-28
(ILLY033,11:26:07/0YA2)

 ***** DIAGNOSTIC SUMMARY OF BGSIA

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

 ***** END OF DIAGNOSTIC SUMMARY
 ***** NORMAL END   BGSIA       (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:07
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/MODIFY-JOB-SWITCHES ON=(1,4)
/START-LMS
//MODIFY-LOGGING-PARAMETERS LOG=*MAX
```

```
//OPEN-LIBRARY LIB=REISEN.HASHLIB,MODE=*UPDATE
  LIBRARY IS CLEARED AND PREPARED
//ADD-ELEMENT FROM-FILE=*OMF,TO-ELEM=*LIB-ELEM(TYPE=R),WRITE-MODE=*ANY

INPUT  OMF
OUTPUT LIBRARY= :SQL2:$XXXXXXXX.REISEN.HASHLIB
       ADD UDSHASH AS (R)UDSHASH/@(0001)/2017-06-28
//SHOW-ELEM-ATTR

INPUT  LIBRARY= :SQL2:$XXXXXXXX.REISEN.HASHLIB
TYP NAME    VER (VAR#) DATE
(R) UDSHASH @   (0001) 2017-06-28
      1 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
 //END
```

3.2.4.1 SIA-Protokoll

Das SIA-Protokoll, welches durch DISPLAY ausgedruckt wird, ist fast identisch mit dem Protokoll, das das Dienstprogramm BPSIA ausdruckt. An einigen Stellen enthält es noch nicht die endgültigen Werte, da einige Werte erst von BFORMAT eingetragen werden. Das Protokoll ist ausführlich beschrieben im Handbuch „Sichern, Informieren und Reorganisieren“, SIA PRINT REPORT.

3.2.4.2 Beschreibung des ESTIMATE-REPORT

Beim Protokoll zum BGSIA-Lauf erhalten Sie nach der Startmeldung den ESTIMATE-REPORT. Er dient dazu, die Größe der Benutzerrealms zu schätzen.

Dies ist notwendig, weil z.B. bei zu geringer Größe eines Benutzerrealms BFORMAT keine Formatierung durchführt, falls der betreffende Realm nicht automatisch erweiterbar ist. Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme.

Der ESTIMATE-REPORT gibt immer aus:

- Realm-Nummer
- Realm-Name
- Realm-Größe (in Datenbankseiten).
  Bei der Realm-Größe ist zu beachten, dass der vom ESTIMATE-REPORT ausgegebene Wert je nach Seitenformat der Datenbank (2-Kbyte-/4-Kbyte-/8-Kbyte-Format) unterschiedlich zu interpretieren ist und z.B. nicht direkt in das CREATE-FILE-Kommando übernommen werden darf. So wird im CREATE-FILE-Kommando der Wert für „SPACE=“ in Einheiten von 2K (BS2000-Halfpages) angegeben. Der ESTIMATE-REPORT hingegen liefert Größenangaben in der Einheit „Datenbankseiten“. Für die Umrechnung in 2K-Einheiten sind somit die Größenangaben des ESTIMATE-REPORTS bei einer 4-Kbyte-Datenbank mit dem Faktor 2 und bei einer 8-Kbyte-Datenbank mit dem Faktor 4 zu multiplizieren.

In folgenden Fällen gibt der ESTIMATE-REPORT zusätzliche Angaben aus:

- Im Realm sind Sätze vorhanden, für die in der SSL die COMPRESSION-Klausel angegeben wurde.
- Im Realm sind Sätze mit variablem Datenfeld vorhanden.
- Es wurden SEARCH-Key-Tabellen mit DUPLICATES ALLOWED und TYPE IS DATABASE-KEY-LIST angelegt.

In einer Korrekturtabelle werden Richtwerte ausgegeben. Diese Richtwerte benötigen Sie zur Korrektur, wenn bei Ihren Sätzen nicht so viel Prozent eingespart wird, wie der ESTIMATE-REPORT als Standard annimmt.

In der folgenden Tabelle sind alle Angaben aufgeführt und die Variablen erklärt.

Angaben im ESTIMATE-REPORT	Erklärung der Variablen
`**** FOR USER-REALM` *realm-ref* `NAME IS:` *realmname*	Realm-Nummer; Realm-Name
`A SIZE OF` *size* `BLOCKS WAS ESTIMATED`	Realm-Größe in Datenseiten; dient als Richtwert für die Speichergröße des benannten Benutzerrealm
`** THE RECORD` *rec-ref* `NAME IS` *satzname*	Satzartnummer; Name der Satzart, für die die COMPRESSION-Klausel gilt
`WITH * COMPRESSION * WAS CALCULATED WITH A PROFIT OF 50%`	Bei der Berechnung der Realm-Größe wurde angenommen, dass die Einsparung durch COMPRESSION bei der genannten Satzart 50% beträgt.

`CORRECTION-TABLE:`
`0% 25% 75%`
n`+ `*n*`+ `*n*`-`
	Korrekturtabelle (nur bei COMPRESSION);
	n benennt die Anzahl der Datenseiten, die bei 0% / 25% / 75% Einsparung zur Realm-Größe *größe* hinzugefügt (+) oder abgezogen (-) werden muss.

`**** IN SET` *set-ref* `NAME IS:` *setname* `FOR RECORD` *rec-ref*	Setnummer; Setname; Satzartnummer;
`A SEARCH-KEY-TABLE TYPE * DATABASE-KEY-LIST * WAS CALCULATED WITH 50% DUPLICATES`	Bei der Berechnung der Größe der DATABASE-KEY-LIST wurde davon ausgegangen, dass 50% der Schlüsselwerte Duplikate sind.

`CORRECTION-TABLE:`
`0% 75% 90%`
n`+ `*n*`- `*n*`-`
	Korrekturtabelle;
	n benennt die Anzahl der Datenseiten, die unter Annahme, dass 0% / 75% / 90% der Schlüsselwerte Duplikate sind, zur Realm-Größe *größe* zugefügt oder davon abgezogen werden muss.

Tabelle 14: Variablen im ESTIMATE-REPORT

Die vorgeschlagenen Realm-Größen dienen als Anhaltspunkt, um die Größenordnung zu bestimmen. Sie können aus folgenden Gründen ungenau sein:

- SSL-Angaben für Mengen (DBTT, RECORD-POPULATION, SET-POPULATION) treffen nicht zu.
- Die Einsparung ist nicht genau voraussagbar (z.B. bei Satzarten, für die die COMPRESSION-Klausel gilt oder die ein variables Datenfeld besitzen).
- Bei SEARCH-Key-Tabellen mit DUPLICATES ALLOWED,
  TYPE IS DATABASE-KEY-LIST ist die Anzahl der Schlüsselduplikate unbekannt.
- Bei LOCATION MODE IS CALC oder bei CALC-SEARCH-Keys kann die Anzahl der Überlaufseiten nicht vorausberechnet werden.
- Wegen gemischter Speicherung ist die Größe des ungenutzten Speicherplatzes nicht kalkulierbar.
- Die Reihenfolge der Speicherung beeinflusst die Größe bei Tabellen.
- INCREASE wird nicht berücksichtigt.

Der ESTIMATE-REPORT geht von Maximalwerten für Sätze aus. Da diese Werte am Anfang kaum erreicht werden, können Benutzerrealms zuerst kleiner eingerichtet werden. Später können Sie sie mit dem Dienstprogramm BREORG (siehe Handbuch „Sichern,Informieren und Reorganisieren“) erweitern.

Sie können die Realms auch so einrichten, dass sie bei Bedarf online erweitert werden können (siehe Handbuch „Datenbankbetrieb"; Online-Realm-Erweiterung).

Beispiel

```
ESTIMATE-REPORT

********** FOR USER-REALM     3 NAME IS : AUFTRAGSRLM
      A SIZE OF          52 BLOCKS WAS ESTIMATED

********** FOR USER-REALM     4 NAME IS : BESTELLRLM
      A SIZE OF         77 BLOCKS WAS ESTIMATED

********** FOR USER-REALM     5 NAME IS : KLEIDUNG
      A SIZE OF         67 BLOCKS WAS ESTIMATED
      *** THE RECORD     8 NAME IS : ARTIKELBESCHR
           WITH *COMPRESSION* WAS CALCULATED WITH A PROFIT OF 50%
           CORRECTION-TABLE :     0%       25%     75%
                                  17+       8+      8-

      **** IN SET    28 NAME IS : SYS_RATENZAHLUNG
           FOR RECORD     5 NAME IS : RATENZAHLUNG
           A SEARCH-KEY-TABLE TYPE *DATABASE-KEY-LIST*
           WAS CALCULATED WITH 50% DUPLICATES
           CORRECTION-TABLE:      0%       75%     90%
                                  0+        0-      0-

********** FOR USER-REALM     6 NAME IS : HAUSHALT
      A SIZE OF         32 BLOCKS WAS ESTIMATED
      *** THE RECORD     8 NAME IS : ARTIKELBESCHR
           WITH *COMPRESSION* WAS CALCULATED WITH A PROFIT OF 50%
           CORRECTION-TABLE :     0%       25%     75%
                                  8+        4+      4-
.
.
.
      **** IN SET    12 NAME IS : LIEFERBARE-ARTIKEL
           FOR RECORD     9 NAME IS : ARTIKEL
           A SEARCH-KEY-TABLE TYPE *DATABASE-KEY-LIST*
           WAS CALCULATED WITH 50% DUPLICATES
           CORRECTION-TABLE:      0%       75%     90%
                                  4+        1-      0-
.
.
.
********** FOR USER-REALM    11 NAME IS : ARTIKELRLM
     A SIZE OF         79 BLOCKS WAS ESTIMATED
END OF ESTIMATE-REPORT
```

3.3 Benutzerrealms formatieren mit BFORMAT

Zum Formatieren der Benutzerrealms müssen Sie das Dienstprogramm BFORMAT ablaufen lassen. BFORMAT

- erweitert die SIA um Informationen über die Benutzerrealms,
- legt in jedem Realm eine ACT-KEY-0-Seite, eine ACT-KEY-N-Seite sowie mindestens eine FPA-Seite an und formatiert DBTT- und CALC-Seiten,
- hinterlegt im DBDIR den Hinweis „BFORMAT EXECUTED“.

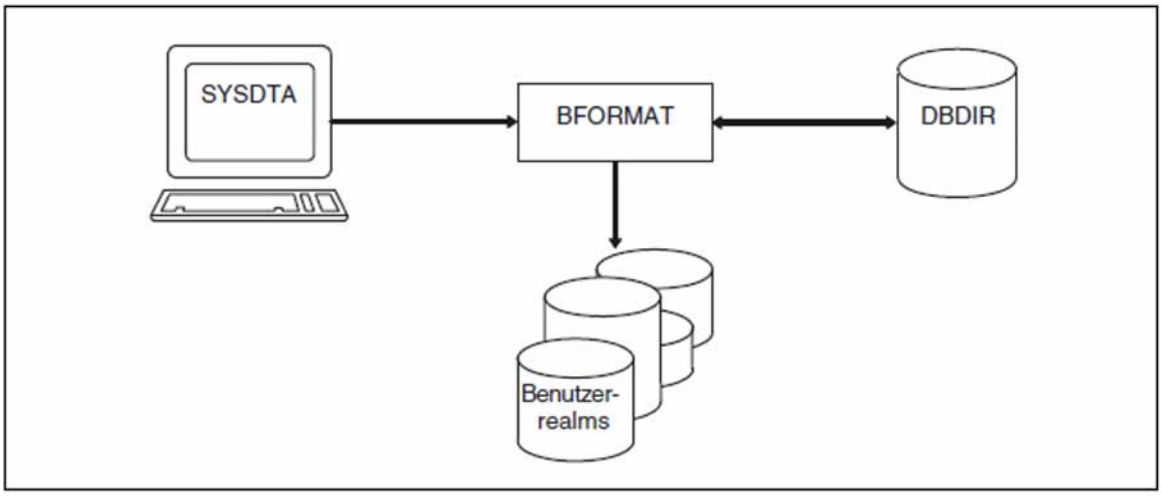

Bild 12: Systemumgebung von BFORMAT

BFORMAT muss in der Kennung aufgerufen werden, unter der die Datenbank katalogisiert ist.

Falls die Benutzerrealms noch nicht eingerichtet wurden, müssen Sie sie spätestens vor dem BFORMAT-Lauf einrichten (siehe Abschnitt „Benutzerrealms einrichten").

BFORMAT erweitert bei Bedarf automatisch die Realms der bearbeiteten Datenbank (sofern die Realms erweiterbar sind). Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme.

BFORMAT berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

Anweisungen für BFORMAT

Die REALM-Anweisung von BFORMAT benennt die Realms, die formatiert werden sollen. Realms können in mehreren BFORMAT-Läufen formatiert werden.
Jeder Realm kann nur einmal formatiert werden.

Erst wenn alle Realms formatiert sind, können Sie den Datenbankaufbau fortsetzen.

Anweisung	Standardwert	Bedeutung
REALM NAME IS { ALL [EXCEPT *realmname1* ,..] | *realmname1* ,.. }	ALL	wahlweise; die angegebenen Realms sollen formatiert /nicht formatiert werden; ALL alle Realms, die in der Schema-DDL definiert wurden, sollen formatiert werden ALL EXCEPT *realmname* Negativliste, d.h. alle Realms, außer dem (den) benannten Realm(s), sollen formatiert werden *realmname* bezeichnet einen Benutzerrealm
END	-	generell erforderlich; beendet die Eingabe der Anweisungen

Tabelle 15: Anweisungen für BFORMAT

> Es ist sinnvoll, Realms einzeln zu formatieren. Bei mehreren Realms betrifft ein Abbruch von BFORMAT auf Grund eines Betriebssystemfehlers ohne Endebehandlung auch bereits erfolgreich formatierte Realms. Der BFORMAT-Lauf muss dann auch für diese Realms wiederholt werden.
>
> Der BFORMAT-Lauf ist sehr schnell, da nur Hashbereiche, FPA- und DBTT-Seiten formatiert werden.

Kommandofolge zum Starten von BFORMAT

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 /CREATE-FILE FILE-NAME=dbname.realmname ...
02 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
03 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
04 /START-UDS-BFORMAT
05 [bformat-sanweisung]
06 END
```

01 siehe Abschnitt „Benutzerrealms einrichten".

03 Die angegebene Version von BFORMAT wird ausgewählt. Die Angabe der Version wird generell empfohlen, da mehrere UDS/SQL-Versionen parallel installiert sein können.

04 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen BFORMAT gestartet werden.

05 Wenn Sie die REALM-Anweisung nicht angeben, werden alle Realms formatiert.

Beispiel

```
/CREATE-FILE FILE-NAME=REISEN.REISENRLM,SUPPORT=PUBLIC-DISK(SPACE=RELATIVE -
/   (PRIMARY-ALLOCATION=220,SECONDARY-ALLOCATION=60))
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=REISEN.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BFORMAT
***** START       BFORMAT     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:07
 * VERSION RECORDS EXPANDED
 REALM NAME IS ALL
 END
 * REISENRLM SUCCESSFULLY FORMATTED
 * SUCHFRAGENRLM INITIALISED IN DBDIR
 * BFORMAT-CONTROL-RECORD WRITTEN TO DBDIR
 ***** ALL REALMS FORMATTED

 ***** DIAGNOSTIC SUMMARY OF BFORMAT

              NO WARNINGS
              NO ERRORS
              NO SYSTEM-ERRORS

 ***** END OF DIAGNOSTIC SUMMARY
 ***** NR OF DATABASE ACCESSES  :         120
 ***** NORMAL END   BFORMAT     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:11
```

3.4 Subschema generieren

Zum Generieren eines Subschemas müssen Sie folgende Programme ablaufen lassen:

DDL-Compiler	Übersetzen der Subschema-Beschreibung
BGSSIA	Erzeugen der Subschema Information Area (SSIA).

3.4.1 Subschema-DDL übersetzen

Die Subschema-DDL (SDDL) wird von demselben DDL-Compiler übersetzt wie die Schema-DDL. Die SDDL-Compiler-Anweisungen sind in Tabelle 12 ("Schema-DDL übersetzen") beschrieben.

Sie müssen dem DDL-Compiler die Subschema-DDL als Eingabedatei zuweisen. Nach dem Übersetzungslauf wird die übersetzte Subschema-Beschreibung im DBCOM (Database Compiler Realm) abgelegt. Mit Hilfe dieser Information baut BGSSIA später die SSIA im DBDIR (Database Directory) auf. Außerdem speichert der DDL-Compiler das aus der übersetzten Subschema-Beschreibung abgeleitete, transformierte Subschema im COSSD und legt eine Prüftabelle (CHECK-TABLE) zu diesem Subschema an. Diese Angaben benötigt der COBOL-Compiler für die Syntax- und Semantikprüfungen der DML-Anweisungen.
Bei der Übersetzung eines Subschemas, das auch in KDBS-Anwendungen genutzt werden soll, müssen Sie die DDL-Compiler-Anweisung „SUBSCHEMA FORM IS OLD“ (Tabelle 12, "Schema-DDL übersetzen") angeben. Der DDL-Compiler erstellt dann das transformierte Subschema und die CHECK-TABLE im Format der UDS/SQL V1.2 („altes“ Format mit 1 byte langen Referenznummern für Satzarten und Sets; siehe Tabelle 12, "Schema-DDL übersetzen").

Das COSSD dient außerdem als Eingabe für das Dienstprogramm BCALLSI (siehe Abschnitt "Zusätzliche Maßnahmen bei CALL-DML-Programmen mit BCALLSI"). BCALLSI erzeugt das SSITAB-Modul, das die Subschema-Informationen für CALL-DML-Programme bereitstellt.

Nach dem Übersetzen der Subschemata sollte der Datenbankadministrator eine Sicherung der Datenbank erstellen (siehe Abschnitt „Datenbank sichern"). Dadurch wird sichergestellt, dass ein konsistenter Sicherungsstand der Datenbank vorhanden ist.

> ¡ Durch das Subschema wird der Satzbereich (RECORD AREA) festgelegt.
> Die Länge des Satzbereichs ist die Summe der Längen aller im zu Grunde liegenden Subschema enthaltenen Satzarten (auf Doppelwortgrenze ausgerichtet) und aller implizit definierten Datenfelder, d.h. ALIAS-Felder und AREA-IDs bei verteilten Satzarten.
> Der DDL-Compiler bricht die Übersetzung des Subschemas mit Fehler ab, sobald der zugehörige Satzbereich größer wird als 65 535 byte (bzw. größer als 61 328 byte bei Angabe von SUBSCHEMA FORM IS OLD).

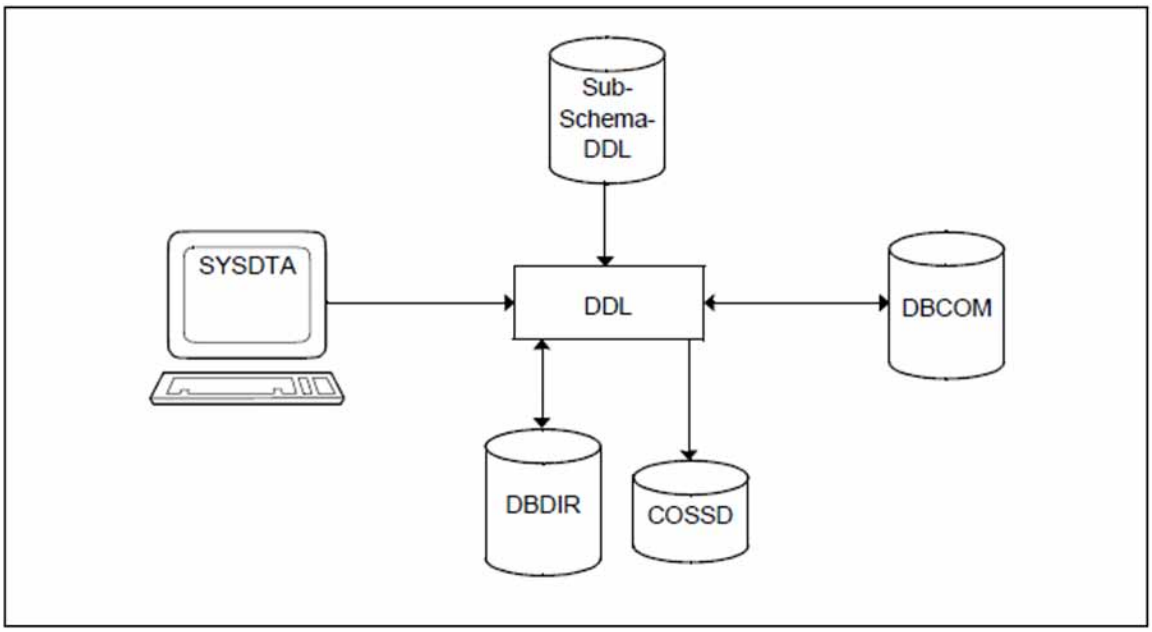

Bild 13: Systemumgebung bei der Subschema-Übersetzung

Kommandofolge zum Übersetzen des Subschemas

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /START-UDS-DDL
04 sddl-compiler-anweisungen
05 END
```

02 Versionsabhängiges Modul des linked-in DBH der entsprechenden Version wird nachgeladen (siehe Handbuch „Anwendungen programmieren“, Abschnitt „UDS/SQL-TIAM-Anwenderprogramme binden, laden und starten“).

03 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen DDL gestartet werden.

04 Die einzelnen Anweisungen können durch Kommas oder Leerzeichen getrennt in einer Zeile eingegeben werden.

Beispiel

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=REISEN.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.9B00
/START-UDS-DDL
***** START        DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:11
*  DDLCOMP: INPUT SYSTEMPARAMETERS
SOURCE IS 'S.REISEN.SUBDDL'
END
*  DDLCOMP: READ SCHEMA/SUBSCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:26:11/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:26:11/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.PUBS
0YA2: PUBSETS:        SQL2
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  DDLCOMP: START SUBSCHEMA-PHASE
*  DDLCOMP: CHECK SUBSCHEMA RULES
*  DDLCOMP: CHECK DATA ALLOCATION
*  DDLCOMP: SUBCOPY
*  DDLCOMP: ERROR DIAGNOSTIC
*  DDLCOMP: NO ERRORS IN SUBSCHEMA-PHASE
*  DDLCOMP: WRITE SUBSCHEMA ON COSSD
*  DDLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:26:11/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: REISEN                 3011       5388         77       1249         57
%  UDS0213 UDS NORMAL BEENDET MIT ************3011 DML-STATEMENTS 2017-06-28
(ILLY033,11:26:11/0YA2)

***** DIAGNOSTIC SUMMARY FOR DDL-SUBSCHEMA

             NO ERRORS
             NO WARNINGS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:11
```

3.4.2 Subschema Information Area (SSIA) erzeugen mit BGSSIA

Die übersetzte Subschema-Beschreibung liegt nach dem Übersetzungslauf durch den DDL-Compiler im DBCOM vor. Diese übersetzte Beschreibung benötigt das Dienstprogramm BGSSIA, um die Subschema Information Area (SSIA) zu erzeugen.

Jede SSIA wird im DBDIR als Satz der internen Satzart SSIA-RECORD gespeichert.
Eine SSIA enthält Informationen des Subschemas, die der DBH braucht, um auf die Datenbank zuzugreifen. Sie können die SSIA mit der DISPLAY-Anweisung von BGSSIA oder mit dem Dienstprogramm BPSIA ausdrucken lassen (siehe Handbuch „Sichern, Informierenund Reorganisieren", SSIA PRINT REPORT).

BGSSIA erweitert bei Bedarf automatisch DBDIR und DBCOM der bearbeiteten Datenbank bzw. die DBTTs der Satzarten im DBCOM. Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme).

BGSSIA berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

Während des Ablaufs arbeitet das Dienstprogramm BGSSIA mit dem linked-in DBH.

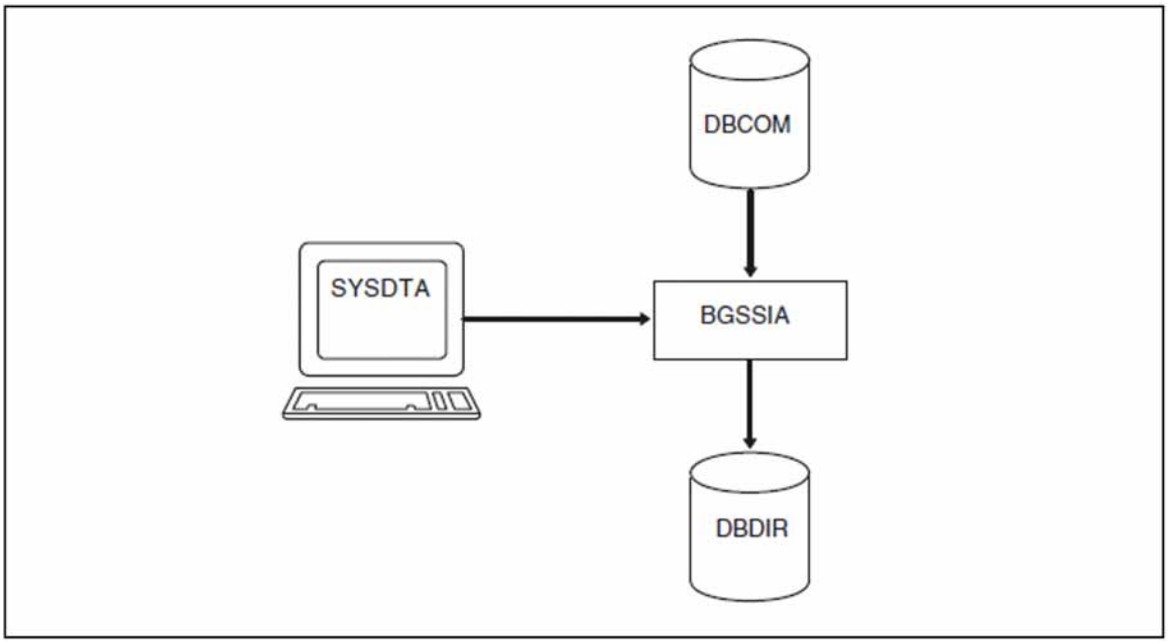

Bild 14: Systemumgebung von BGSSIA

Anweisungen für BGSSIA

Anweisung	Standardwert	Bedeutung
GENERATE SUBSCHEMA *subschemaname* OF SCHEMA *schemaname*	-	wahlweise; *subschemaname*: Name des Subschemas *schemaname*: Name des Schemas Prüfen, ob bereits eine SSIA für ein bestimmtes Subschema vorhanden ist und Erstellen von: SSIA mit einzelnen Informationen über Realms, Satzarten und Sets Listen der einzelnen Felder Listen aller im Subschema auftretenden Namen
DELETE SUBSCHEMA *subschemaname* OF SCHEMA *schemaname*	-	wahlweise; Löschen einer früher erzeugten SSIA aus dem DBDIR
REGENERATE SUBSCHEMA *subschemaname* OF SCHEMA *schemaname*	-	wahlweise; Löschen der alten SSIA und neu Erstellen der SSIA (Funktion DELETE und GENERATE zusammengefasst); geeignet zum Korrigieren eines Subschemas
DISPLAY[SUBSCHEMA *subschemaname* OF SCHEMA *schemaname*]	-	wahlweise; gilt nur in Verbindung mit der GENERATE-Anweisung oder REGENERATE-Anweisung: Ausdrucken der SSIA; die Angabe DISPLAY genügt
END	-	generell erforderlich; beendet die Eingabe der Anweisungen

Tabelle 16: Anweisungen für BGSSIA

Kommandofolge zum Starten von BGSSIA

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR

02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK

03 /START-UDS-BGSSIA

04 bgssia-anweisungen

05 END
```

02 Versionsabhängiges Modul des linked-in DBH der entsprechenden Version wird nachgeladen (siehe Handbuch „Anwendungen programmieren", Abschnitt „UDS/SQL-TIAM-Anwenderprogramme binden, laden und starten“).

03 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen BGSSIA gestartet werden.

Beispiel

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=REISEN.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BGSSIA
 ***** START        BGSSIA       (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:11
 GENERATE SUBSCHEMA REISEBUCHUNG OF SCHEMA REISEBUERO
 DISPLAY
 END
 %  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:26:11/0YA2)
 %  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:26:11/0YA2)
 0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.PUBS
 0YA2: PUBSETS:        SQL2
 0YA2: DEFAULT PUBSET: SQL2
 0YA2: ----------------------------------------
 *** SSIA GENERATION NORMALLY ENDED.
 *GENERATION OF ITEM-TABLE AND NAME-TABLE STARTED.
 *GENERATION OF ITEM-TABLE AND NAME-TABLE FINISHED.
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:26:11/0YA2)
 0YA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: REISEN                  1800       2529         76        296         27
 %  UDS0213 UDS NORMAL BEENDET MIT ************1800 DML-STATEMENTS 2017-06-28
 (ILLY033,11:26:11/0YA2)

 ***** DIAGNOSTIC SUMMARY OF BGSSIA

              NO WARNINGS
              NO ERRORS
              NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END BGSSIA (UDS/SQL V2.9 0000 ) 2017-06-28 11:26:11
```

3.5 Zusätzliche Maßnahmen bei CALL-DML-Programmen mit BCALLSI

Das Dienstprogramm BCALLSI müssen Sie ablaufen lassen, wenn Sie CALL-DML-Programme haben oder mit DMLTEST (CALL-DML-Anwendungen) arbeiten.

BCALLSI erzeugt das SSITAB-Modul (SUBSCHEMA INFORMATION TABLE) mit den Subschema-Informationen, die ein CALL-DML-Programm zum Zeitpunkt des Programmlaufs braucht.

BCALLSI berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

Subschema-Informationen bereitstellen

Zum Abarbeiten von DML-Anweisungen benötigt der DBH Informationen über das jeweilige Subschema. Folgende Informationen liegen, vom DDL-Compiler hinterlegt, im COSSD vor:

- das transformierte Subschema
- die sog. Prüftabelle (CHECK-TABLE)

Für den DBH werden diese Informationen auf unterschiedlichen Wegen zusammengestellt:

- Bei COBOL-DML-Programmen braucht der COBOL-Compiler die Subschema-Information zum Zeitpunkt der Übersetzung des Anwenderprogramms.
- Bei CALL-DML-Programmen wird die Subschema-Information zum Zeitpunkt des Programmlaufs benötigt. Da der Zugriff auf das COSSD zum Zeitpunkt des Programmlaufs zu zeitaufwendig wäre, erzeugen Sie zunächst mit BCALLSI das SSITAB-Modul, wobei BCALLSI die Informationen des COSSD verwendet. Zum Zeitpunkt des Programmlaufs wird das SSITAB-Modul aus der Modulbibliothek vom CALL-DML-Verbindungsmodul nachgeladen. Bei CALL-DML-Programmen muss demnach der BCALLSI-Lauf zusätzlich zwischen dem Übersetzen der Subschema-DDL und dem Ablauf des Programms erfolgen.

Spezielle Subschemata im „alten“ Format bearbeiten

Neben dem Standardformat, in dem das transformierte Subschema und die zugehörige Prüftabelle in der COSSD vorliegen, wird auch das "alte" Format bis einschließlich UDS/SQL V1.2 mit 1 byte langen Referenznummern für Satzarten und Sets (siehe auch Tabelle 12, "Schema-DDL übersetzen") noch durch BCALLSI bearbeitet. Das "alte" Format wird für Subschemata benötigt, die in KDBS-Anwendungen bearbeitet werden . Ein COSSD kann gleichzeitig transformierte Subschemata im Standardformat wie im "alten" Format enthalten. Aus einem transformierten Subschema im "alten" Format erzeugt BCALLSI ein SSITAB-Modul im Format der UDS/SQL V1.2, das weiterhin vom aktuellen CALL-DML-Umsetzer unterstützt wird.

Funktionen von BCALLSI

BCALLSI kann sowohl auf ein COSSD einer UDS/SQL Version > V1.2 als auch auf ein COSSD der UDS/SQL V1.2 oder UDS/SQL V1.1 zugreifen.

BCALLSI erfüllt folgende Funktionen:

- Übersetzen des transformierten Subschemas in Realm-, Set-, Satz- und Feldtabellen.
- Ausdrucken des transformierten Subschemas.
- Prüfen der Realm-, Set-, Satz- und Feldnamen auf Eindeutigkeit innerhalb der ersten acht bzw. 30 Zeichen. Sind die Namen nicht eindeutig, so wird im Anschluss an die Liste des transformierten Subschemas eine Warnung ausgegeben.
- Kopieren der Prüftabelle aus dem COSSD zum Vervollständigen der SSITAB.

- Ausgeben des SSITAB-Moduls in die EAM-Datei unter dem Namen *subschema*##, wobei *subschema* aus den ersten sechs Zeichen des vollständigen Subschemanamens gebildet wird.

Das erzeugte SSITAB-Modul müssen Sie anschließend mit Hilfe des BS2000-Dienstprogramms LMS in eine Bibliothek eintragen. Der Name der Bibliothek ist frei wählbar. Der DBH lädt vorrangig SSITAB-Module aus einer Bibliothek nach, die mit dem Linknamen $UDSSSI zugewiesen wird. Wenn die SSITAB-Module nicht nur in einer Bibliothek gehalten werden, z.B. pro Datenbank in einer eigenen, können die weiteren Bibliotheken mit den Linknamen BLSLIB00 bis BLSLIB99 zugewiesen werden (siehe Handbuch „Datenbankbetrieb", Abschnitt DBH-Startkommandos und Handbuch „Anwendungen programmieren", Abschnitt UDS/SQL-TIAM-Anwendungen binden, laden und starten).

> ¡ Subschemanamen müssen in den ersten sechs Zeichen eindeutig sein, da der Name des SSITAB-Moduls aus den ersten sechs Zeichen plus '##' gebildet wird.

Systemumgebung von BCALLSI

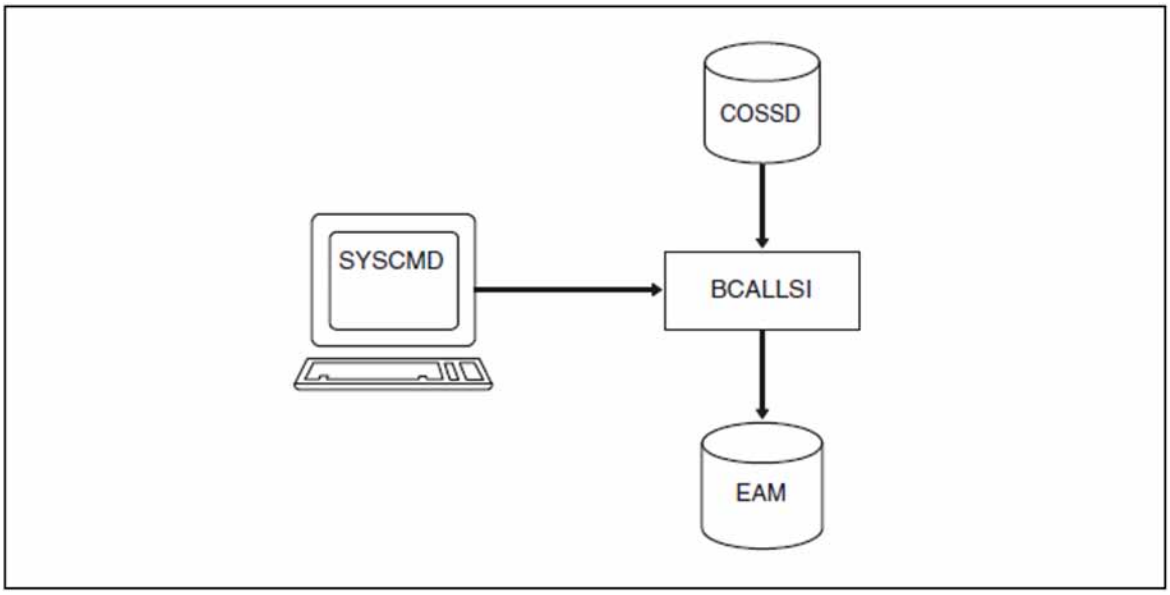

Bild 15: Systemumgebung von BCALLSI

Anweisungen für BCALLSI

Anweisungen	Standard-wert	Bedeutung
{SCHEMA | S}=*schemaname*, {SUBSCHEMA | SS}=*subschemaname*	-	generell erforderlich; weist BCALLSI die Namen des Schemas und des Subschemas zu *schemaname* Name des Schemas, der in der Schema-DDL vergeben wurde *subschemaname* Name des Subschemas, der in der Subschema-DDL vergeben wurde
[{,MESSAGE | ,M}= {*ALL|N[O-AMBIGUITY-8]}]	*ALL	*ALL Alle Eindeutigskeits-Verletzungen, auch in den ersten 8 Zeichen, werden auf SYSLST einzeln ausgegeben. NO-AMBIGUITY-8 Eindeutigkeits-Verletzungen in den ersten 8 Zeichen eines Namens werden nicht einzeln ausgegeben

Tabelle 17: Anweisungen für BCALLSI

Kommandofolge zum Starten von BCALLSI

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 /DELETE-SYSTEM-FILE FILE-NAME=*OMF
02 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
03 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
04 /START-UDS-BCALLSI
05 bcallsi-anweisung
```

03 Die angegebene Version von BCALLSI wird ausgewählt. Die Angabe der Version wird generell empfohlen, da mehrere UDS/SQL-Versionen parallel installiert sein können.

04 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen BCALLSI gestartet werden.

05 Bei BCALLSI gibt es keine END-Anweisung!

SSITAB-Modul in die Modulbibliothek eintragen

```
01 /START-LMS
02 //OPEN-LIB LIB=modlib,MODE=*UPDATE
```

```
03 //ADD-ELEMENT FROM-FILE=*OMF,TO-ELEMENT=*LIBRARY-ELEMENT(TYPE=R)

04 //END
```

Beispiel

```
/DELETE-SYSTEM-FILE FILE-NAME=*OMF
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=VERSAND.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BCALLSI
 ***** START       BCALLSI     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:28
 SCHEMA=ARTIKELVERSAND,SUBSCHEMA=ADMIN,MESSAGE=*ALL
 WARNING: THERE ARE  NAME AMBIGUITIES IN THE FIRST 8 CHARACTERS OF SOME NAMES
 SEE PRINTOUT!
 SSITAB ADMIN##  WRITTEN TO EAM-OMF

 ***** DIAGNOSTIC SUMMARY OF BCALLSI

               NO WARNINGS
               NO ERRORS
               NO SYSTEM-ERRORS

 ***** END OF DIAGNOSTIC SUMMARY
 ***** NR OF DATABASE ACCESSES  :           0
 ***** NORMAL END   BCALLSI     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:28
/MODIFY-JOB-SWITCHES ON=(1,4)
/START-LMS
//MOD-LOG-PAR LOG=*MAX
//OPEN-LIB LIB=LMS.SSITAB,MODE=*UPD(STATE=*ANY)
//ADD-ELEMENT FROM-FILE=*OMF,TO-ELEM=*LIB(TYPE=R),WRITE-MODE=*ANY
INPUT  OMF
OUTPUT LIBRARY= :SQL2:$XXXXXXXX.LMS.SSITAB
       ADD UDSHASH AS (R)UDSHASH/@(0002)/2017-06-28 , OUTPUT REPLACED
       ADD ADMIN## AS (R)ADMIN##/@(0001)/2017-06-28

//SHOW-ELEM-ATTR ELEM=*LIB-ELEM()

INPUT  LIBRARY= :SQL2:$XXXXXXXX.LMS.SSITAB
TYP NAME     VER (VAR#) DATE        NAME     VER (VAR#) DATE
(R) ADMIN##  @   (0002) 2017-06-28  UDSHASH  @   (0002) 2017-06-28
(R) VERWAL## @   (0004) 2017-06-28
       3 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//END

PRINTOUT:
    .
    .
WARNING:    ITEM NAME  'KUNDEN-N'  OF RECORD TYPE  'KUNDE   '  IS   2-FOLD AMBIGUOUS.
WARNING:    ITEM NAME  'AUFTR-PO'  OF RECORD TYPE  'AUFTR-PO'  IS   2-FOLD AMBIGUOUS.
WARNING:    ITEM NAME  'NEXT-RAT'  OF RECORD TYPE  'RATENZAH'  IS   3-FOLD AMBIGUOUS.
WARNING:  RECORD NAME  'ARTIKELA'                          IS   2-FOLD AMBIGUOUS.
WARNING:    ITEM NAME  'LIEFER-N'  OF RECORD TYPE  'LIEFERAN'  IS   2-FOLD AMBIGUOUS.
WARNING:    ITEM NAME  'LIEFER-P'  OF RECORD TYPE  'LIEFERAN'  IS   2-FOLD AMBIGUOUS.
WARNING:    ITEM NAME  'LIEFER-S'  OF RECORD TYPE  'LIEFERAN'  IS   2-FOLD AMBIGUOUS.
    .
    .
             SCHEMANAME       :  ARTIKELVERSAND
             SUBSCHEMANAME    :  ADMIN
             MODUL-ENTRY      :  ADMIN##
```

```
              LENGTH OF MODUL  :          6408 BYTES
              SSITAB-VERSION   :             2
```

4 Zugriffsberechtigungen festlegen (ONLINE-PRIVACY, BPRIVACY)

In UDS/SQL stehen Ihnen mit ONLINE-PRIVACY und BPRIVACY zwei Dienstprogramme zur Verfügung, mit denen Sie festlegen können, wer (Benutzergruppen) auf welche Art und Weise (Zugriffsrechte) auf eine Datenbank zugreifen darf.

Funktionsumfang sowie Syntax und Semantik der Rechtevergabe sind bei ONLINE-PRIVACY und BPRIVACY identisch.

- ONLINE-PRIVACY setzen Sie online ein, d. h. während die Datenbank zu einer Independent-DBH-Session zugeschaltet ist. Sie können so die Zugriffsberechtigungen für eine Datenbank im laufenden Betrieb abfragen oder ändern.
- BPRIVACY setzen Sie offline ein, z. B. beim Datenbankaufbau, um die Zugriffsberechtigungen für die Datenbank festzulegen.

ONLINE-PRIVACY und BPRIVACY bieten folgende Funktionen:

- Benutzergruppen mit oder ohne Zugriffsrechte definieren
- Benutzergruppen löschen
- den Benutzergruppen Zugriffsrechte erlauben oder verbieten
- Informationen über die Benutzergruppen ausgeben

Bei Zusammenarbeit mit *open*UTM kann der Zugriffsschutz von UDS/SQL gemeinsam mit dem Zugriffsschutz von *open*UTM eingesetzt werden (siehe *open*UTM-Handbuch „Anwendungen generieren“).

4.1 Benutzergruppen

Unter Benutzergruppen versteht man Personenkreise, die auf die Datenbank zugreifen dürfen.

Der Name einer Benutzergruppe setzt sich im Allgemeinen wie folgt zusammen (genaueres siehe Tabelle 18 im "Rechteprüfung"):

host + *appl* + *grp*

Dabei bedeutet:

host	Name des Verarbeitungsrechners
appl	Name der UDS/SQL/*open*UTM-Anwendung bzw. „_“
grp	BS2000-Kennung oder bei einer UDS/SQL-*open*UTM-Anwendung der KSET-Name, der der *open*UTM-Benutzerkennung zugeordnet ist.

Die drei Einzelbestandteile werden intern von UDS/SQL jeweils mit Unterstrich „_“ auf acht Zeichen aufgefüllt.

Die entsprechenden Informationen werden von *open*UTM oder UDS/SQL ermittelt. Daher sind keine Angaben mehr über die Zugriffsrechte in den Anwendungsprogrammen notwendig (z.B. PRIVACY-RECORD in der SUB-SCHEMA SECTION, PERMIT in SQL-Programmen). UDS/SQL ignoriert evtl. vorhandene Angaben in bestehenden Programmen.

Den Benutzergruppennamen und die Zugriffsrechte müssen Sie mit ONLINE-PRIVACY bzw. mit BPRIVACY definieren, bevor die entsprechende Benutzergruppe Datenbankaufrufe absetzen kann.

Weitere Informationen finden Sie im Handbuch „Datenbankbetrieb“, das ein Beispiel enthält, wie Sie UDS/SQL-Benutzergruppen definieren und Zugriffsrechte vergeben.

4.2 Zugriffsrechte

Unter Zugriffsrechten versteht man Zugriffe auf Datenbankobjekte wie Lesen (RETRIEVAL) bzw. Lesen und Ändern (UPDATE). Zugriffsrechte können den Benutzergruppen erlaubt oder wieder entzogen werden.

Einer Benutzergruppe können die Zugriffsrechte Lesen (RETRIEVAL) oder Lesen und Ändern (UPDATE) auf die folgenden Datenbankobjekte erteilt werden.

Zugriffsrechte bei CODASYL-Anwendungen auf:

- Realms
- Satzarten (RECORDs)
- Sets

Zugriffsrechte bei SQL-Anwendungen auf:

- Basistabellen
- Fremdschlüssel

Die Zugriffe RETRIEVAL, UPDATE und ALL können wie folgt vereinbart werden:

- Zugriff erlauben:
  Anweisungen ADD-USER-GROUP/GRANT-ACCESS
- Zugriff wieder entziehen:
  Anweisung REVOKE-ACCESS
- Zugriff differenzieren:
  Anweisungen GRANT-ACCESS/REVOKE-ACCESS

Wenn Zugriffsrechte mit der Anweisung GRANT-ACCESS vergeben werden, so bleiben eventuell schon vorhandene Zugriffsrechte für ein gewisses Datenbankobjekt erhalten und die neu definierten kommen hinzu. Entsprechend werden bei der Anweisung REVOKE-ACCESS nur die bei dieser Anweisung definierten Zugriffsrechte entzogen, d.h. wenn die Benutzergruppe weitere Zugriffsrechte bereits besitzt, bleiben diese bestehen.

4.3 Rechteprüfung

UDS/SQL führt die Rechteprüfung ausschließlich über den Benutzergruppennamen durch.

Der Benutzergruppenname muss mit ONLINE-PRIVACY bzw. mit BPRIVACY definiert und die Zugriffsrechte müssen vergeben worden sein, bevor die entsprechenden Benutzergruppen Datenbankaufrufe absetzen können.

Kann der DBH die Benutzergruppe nicht identifizieren, wird dem Anwenderprogramm ein Statuscode übergeben bzw. wird die IQS-Sitzung beendet.

Der folgenden Tabelle entnehmen Sie, wie die Benutzergruppennamen aufgebaut sind, welche Konfiguration mit welchem Gruppenbegriff überprüft wird und wie Sie die entsprechenden Benutzergruppen in der ADD-USER-GROUP-Anweisung definieren müssen. Die Begriffe „lokal“ und „entfernt“ sind relativ zum Datenbankort zu sehen.

Konfiguration	**Wert**			**Definition in der ADD-USER-GROUP-Anweisung**
	host	**appl**	**grp**	
*open*UTM Anw. ohne KSET	*host*	*appl*	*kset*	*KSET-FORMAT(HOST=*host*,APPLICATION=*appl*, KSET=*NONE)
*open*UTM Anw. mit KSET	*host*	*appl*	*kset*	*KSET-FORMAT(HOST=*host*,APPLICATION=*appl*, KSET=*kset*)
TIAM	*host*	' _'	*Kenn*	*FREE-FORMAT(HOST=*host* ,USER-ID=*Kenn*)
linked-in	*host*	' _'	*Kenn*	*FREE-FORMAT(HOST=*host* ,USER-ID=*Kenn*)

Tabelle 18: Aufbau der Benutzergruppenangaben

Erklärungen

host Name des Verarbeitungsrechners, auf dem die UDS/SQL-openUTM-Anwendung bzw. das UDS/SQL-Anwenderprogramm läuft.
Hier müssen Sie den Standardnamen des eigenen Prozessors aus DCAM-Sicht angeben. Sofern im TIAM-Fall kein DCAM zur Verfügung steht, geben Sie HOST=LOCAL an.

appl Name der openUTM-Anwendung

kset KSET-Name, der der betreffenden openUTM-Anwendung zugeordnet ist

Kenn BS2000-Benutzerkennung

In Anwenderprogrammen (COBOL-DML, CALL-DML, SQL) und bei IQS werden zum Teil noch „alte“ PRIVACY-Benutzerangaben (< UDS/SQL V1.2) gemacht bzw. vorausgesetzt (IQS):

- „Alte“ Angaben im sogenannten PRIVACY RECORD (< UDS/SQL-Version 1.2) in Anwenderprogrammen (COBOL-DML, CALL-DML, SQL) oder bei IQS werden von UDS/SQL ignoriert.
- Die PRIVACY-Angaben bei IQS einer Version <= 3.1 dürfen nicht leer sein, sind aber ansonsten beliebig.

Die Rechteprüfung erfolgt über den Benutzergruppennamen, der sich aus dem Namen des Verarbeitungsrechners und der Ablaufkennung der TIAM-Anwendung bzw. dem Namen der openUTM-Applikation (mit oder ohne KSET-Angabe) zusammensetzt.

Der KSET-Name kann entfallen, wenn bei der entsprechenden openUTM-Anwendung kein KSET-Name vereinbart wurde. Falls keine openUTM-Benutzer definiert sind, wird ein ggf. definierter KSET-Name eines logischen Terminals (LTERM) zur Rechteprüfung verwendet.
openUTM verwendet vordefinierte KSET-Namen, die Sie mit KDCINF KSET anzeigen lassen können. Auch für diese vordefinierten KSET-Namen müssen die Zugriffsrechte auf die Datenbank festgelegt werden.
Bei verteilter Transaktionsverarbeitung (VTV) mit openUTM-D ist der KSET-Name aus dem zugehörigen LPAP-Eintrag zu verwenden (siehe openUTM-Handbuch „Anwendungen generieren").

4.4 Systemumgebung von ONLINE-PRIVACY

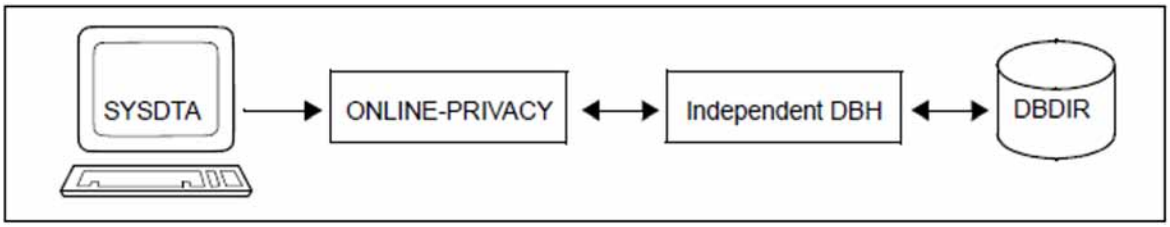

Bild 16: Systemumgebung von ONLINE-PRIVACY

Das Dienstprogramm ONLINE-PRIVACY läuft als UDS/SQL-TIAM-Anwendungsprogramm einer Independent-DBH-Session.

Sie können ONLINE-PRIVACY zu jedem Zeitpunkt während des laufenden Betriebs einer Datenbank starten, wenn folgende Voraussetzungen gegeben sind:

- ONLINE-PRIVACY wird in der Kennung aufgerufen, unter der die Datenbank katalogisiert ist.
- Die Datenbank ist zu einer Independent-DBH-Session zugeschaltet.
- Die Datenbank ist nicht im Modus SHARED-RETRIEVAL an die Session angeschlossen.

Mit ONLINE-PRIVACY ist es beliebig oft möglich, neuen Benutzergruppen den Zugang zu einer bestehenden Datenbankanwendung zu erlauben und die Zugriffsrechte bisheriger Benutzergruppen zu ändern.
Nach Änderungen in einem Schema (z.B. neue Satzarten oder neue Basistabellen wurden eingeführt) müssen Sie die Zugriffsrechte aktualisieren.

Durch Aktivierung der Online-Erweiterbarkeit des DBDIR mit ACT INCR können Sie dafür sorgen, dass der DBDIR im Bedarfsfall durch den DBH erweitert wird. Eine Online-Erweiterung der DBTTs der Satzarten des DBDIR findet jedoch nicht statt.

> ¡ Ein Zugriff auf eine Datenbank in einer Remote Konfiguration über UDS-D ist mit ONLINE-PRIVACY nicht möglich.

Zugriffssperren

Eine mit dem DAL-Kommando ACCESS auf Datenbank- oder Realm-Ebene gesetzte Zugriffssperre (ACCESS LOCK) oder Zugriffseinschränkung (ACCESS RETRIEVAL) hat für das Dienstprogramm ONLINE-PRIVACY keine Wirkung.

ONLINE-PRIVACY gibt Ihnen deshalb die Möglichkeit, auf einer Datenbank Rechteänderungen vorzunehmen, bei der nur lesende Transaktionen zugelassen sind. Die Datenbank muss sich hierfür im Zuschaltmodus EXCLUSIVE-UPDATE befinden und mit dem DAL-Kommando ACCESS RETRIEVAL,DB=dbname für ändernde Transaktionen gesperrt sein.

Wirksamkeit der Rechteänderungen

Die Änderungen der Rechtevergabe für eine Datenbank, die Sie mit ONLINE-PRIVACY vornehmen, wirken sich auf alle Verarbeitungsketten aus, die nach Beendigung des ONLINE-PRIVACY-Laufs (Anweisung FINISH der ONLINE-PRIVACY-Transaktion) gestartet werden.

Verarbeitungsketten, die vor dem Ende des ONLINE-PRIVACY-Laufs gestartet wurden, arbeiten weiter mit den alten Privacy-Informationen.

Auswirkungen auf den Communication Pool (CUP)

Das Dienstprogramm ONLINE-PRIVACY schließt sich an den Communication Pool (CUP) an, wie andere UDS /SQL-TIAM-Anwenderprogramme auch. Der Platzbedarf von ONLINE-PRIVACY innerhalb des Communication Pools entspricht in etwa der Größe der SIA der zu bearbeitenden Datenbank.

Gehen Sie wie folgt vor, um sicherzustellen, dass die Größe des Communication Pools für die Verwendung von ONLINE-PRIVACY ausreichend dimensioniert ist:

- Ermitteln Sie die Größe des SIA mit dem Dienstprogramm BPSIA.
- Berücksichtigen Sie die Größe des SIA als zusätzlichen Platzbedarf für ONLINE-PRIVACY bei der Berechnung der Mindestgröße des Communication Pools.
- Versorgen Sie den DBH-Ladeparameter PP CUP-SIZE entsprechend.

4.5 Systemumgebung von BPRIVACY

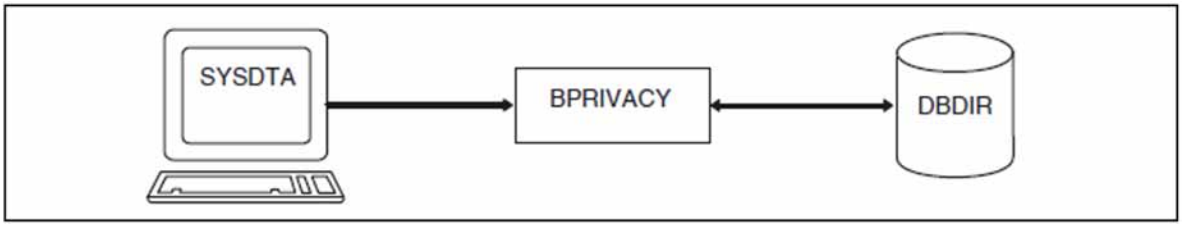

Bild 17: Systemumgebung von BPRIVACY

Das Dienstprogramm BPRIVACY muss in der Kennung aufgerufen werden, unter der die Datenbank katalogisiert ist.
Sie können BPRIVACY zu jedem Zeitpunkt nach Ablauf des Programms BFORMAT starten. Somit ist beliebig oft möglich, neuen Benutzergruppen den Zugang zu einer bestehenden Datenbankanwendung zu erlauben und die Zugriffsrechte bisheriger Benutzergruppen zu ändern.
Nach Änderungen in einem Schema (z.B. neue Satzarten oder neue Basistabellen wurden eingeführt) müssen Sie die Zugriffsrechte aktualisieren.

Bei CODASYL-Zugriff kann BPRIVACY in der Aufbauphase einer UDS/SQL-Datenbank vor dem Übersetzen der Subschemata ablaufen.

Bei SQL-Zugriff können Sie BPRIVACY nach dem Übersetzen und dem Eintragen des relationalen Schemas (UDS /SQL-Subschema) in die Datenbank starten.

BPRIVACY erweitert bei Bedarf automatisch den DBDIR der bearbeiteten Datenbank bzw. die DBTTs der Satzarten im DBDIR. Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme).

BPRIVACY berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

Das Dienstprogramm BPRIVACY nutzt in seinem Ablauf den linked-in-DBH.

4.6 Regeln der Anweisungen

Die Anweisungsformate der Dienstprogramme ONLINE-PRIVACY und BPRIVACY entsprechen den Regeln von SDF (System Dialog Facility, siehe Handbücher „Dialogschnittstelle SDF“ und „Kommandos“).

Falsch eingegebene Anweisungen können Sie korrigieren. Jede richtig angegebene Anweisung können Sie mit der Anweisung UNDO oder mit der inversen Funktion - falls vorhanden - zurücknehmen.

Widersprechen sich die Angaben, so gilt immer die letzte Angabe.

Alle gültigen Anweisungen werden erst nach der Anweisung END ausgeführt.
Ausnahme: Anweisung OPEN-DATABASE

Die Dienstprogramme ONLINE-PRIVACY und BPRIVACY arbeiten immer auf allen Objekten (Realms, Records, Sets) des PRIVACY-Schemas (PRIVACY-AND-IQF-SCHEMA).

4.7 Übersicht der Anweisungen

Anweisung	Bedeutung
ADD-USER-GROUP **USER-GROUP-NAME** = list-poss(6): ***KSET-FORMAT**(...) / ***FREE-FORMAT**(...) ,**OBJECT** = **NONE** / list-poss(6): ***REALM**(...) / ***RECORD**(...) / ***SET**(...)	Benutzergruppe definieren ggf. mit Zugriffsrechten
END	Kommandoeingabe beenden
GRANT-ACCESS **USER-GROUP-NAME** = list-poss(6): ***KSET-FORMAT**(...) / ***FREE-FORMAT**(...) ,**OBJECT** = list-poss(6): ***REALM**(...) / ***RECORD**(...) / ***SET**(...)	Zugriffsrechte für eine Benutzergruppe vergeben
OPEN-DATABASE **DATABASE-NAME** = <dbname>	Datenbank eröffnen
REMOVE-USER-GROUP **USER-GROUP-NAME** = **ALL** / ***ALL-EXCEPT**(...) / list-poss(6): ***KSET-FORMAT**(...) / ***FREE-FORMAT**(...)	Eine oder mehrere Benutzergruppe(n) löschen
REVOKE-ACCESS **USER-GROUP-NAME** = list-poss(6): ***KSET-FORMAT**(...) / ***FREE-FORMAT**(...) ,**OBJECT** = list-poss(6): ***REALM**(...) / *RECORD(...) / ***SET**(...)	Einer Benutzergruppe Zugriffsrechte entziehen
SHOW-USER-GROUP **USER-GROUP-NAME** = **ALL** / ***ALL-EXCEPT**(...) / list-poss(6): ***KSET-FORMAT**(...) / ***FREE-FORMAT**(...) ,**OUTPUT** = list-poss: **SYSLST** / **SYSOUT**	Informationen über eine oder mehrere Benutzergruppe(n) ausgeben
UNDO	Anweisung rückgängig machen

Tabelle 19: Übersicht der Anweisungen

4.7.1 ADD-USER-GROUP (Benutzergruppe definieren ggf. mit Zugriffsrechten)

Mit der Anweisung ADD-USER-GROUP können Sie neue Benutzergruppen definieren. Die dazugehörenden Zugriffsrechte können Sie dabei gleich mitdefinieren.

Der Aufbau der Benutzergruppenangabe ist abhängig von der Umgebung, in der Sie arbeiten, siehe Tabelle 18 ("Rechteprüfung").

Den Namen der Benutzergruppe können Sie in zwei unterschiedlich strukturierten Formen angeben:

*KSET-FORMAT und *FREE-FORMAT

*KSET-FORMAT bietet sich z.B. für den openUTM-Betrieb an. Dort setzt sich der Benutzergruppenname zusammen aus den drei Teilen openUTM-Hostname, openUTM-Anwendungsname und KSET-Name.

Falls Sie für die angegebene Benutzergruppe keine Zugriffsrechte definiert haben, wird nur die Benutzergruppe eingerichtet.
Diese Benutzergruppe besitzt noch keine Zugriffsrechte, d.h. alle Zugriffe auf Datenbankobjekte sind verboten. Sie müssen erst durch nachfolgende Anweisungen GRANT-ACCESS (siehe "GRANT-ACCESS (Zugriffsrechte für eine Benutzergruppe vergeben)") definiert werden.

ADD-USER-GROUP

USER-GROUP-NAME = list-poss(6): ***KSET-FORMAT**(...) / ***FREE-FORMAT**(...)

***KSET-FORMAT**(...)

| **HOST** = <host>

| ,**APPLICATION** = <appl>

| ,**KSET** = **<u>*NONE</u>** / list-poss(30): <kset>

***FREE-FORMAT**(...)

| **HOST** = <host>

| ,**USER-ID** = list-poss(30): <userid> / **<u>*NONE</u>**

,**OBJECT** = **<u>NONE</u>** / list-poss(6): ***REALM**(...) / ***RECORD**(...) / ***SET**(...)

***REALM**(...)

| **NAME** = **<u>*ALL</u>** / ***ALL-EXCEPT**(...) / list-poss(30): <realmname>

| | ***ALL-EXCEPT**(...)

| | **NAME** = list-poss(30): <realmname>

| ,**RIGHT** = **ALL** / **RETRIEVAL**

***RECORD**(...)

| **NAME** = **<u>*ALL</u>** / ***ALL-EXCEPT**(...) / list-poss(30): <recordname>

| | ***ALL-EXCEPT**(...)

| | **NAME** = list-poss(30): <recordname>

| ,**RIGHT** = **ALL** / **RETRIEVAL**

***SET**(...)

```
|  NAME = *ALL / *ALL-EXCEPT(...) / list-poss(30): <setname>
|       |   *ALL-EXCEPT(...)
|       |       NAME = list-poss(30): <setname>
|  ,RIGHT = ALL / RETRIEVAL
```

USER-GROUP-NAME = list-poss(6): *KSET-FORMAT(...) / *FREE-FORMAT(...)
Name der Benutzergruppe.

***KSET-FORMAT(...)**
Angabe der Benutzergruppe.

HOST = <host>
Verarbeitungsrechner der openUTM-Anwendung (siehe *host*, "Benutzergruppen").

APPLICATION = <appl>
Name der openUTM-Anwendung.

KSET = *NONE
Es wird kein KSET-Name angegeben.

KSET = list-poss(30): <kset>

KSET-Name der openUTM-Anwendung.

***FREE-FORMAT(...)**
Angabe der Benutzergruppe.

HOST = <host>
Verarbeitungsrechner der Anwendung.

USER-ID = list-poss(30): <userid>
Kennung der Anwendung.

USER-ID = *NONE
Es wird keine Kennung angegeben. Die Angabe *NONE ist nur noch aus Kompatibilitätsgründen erlaubt. Der entsprechende Benutzergruppenname ist ab UDS/SQL V2.0 nicht mehr nutzbar.

OBJECT = NONE / list-poss(6): *REALM(...) / *RECORD(...) / *SET(...)
Die Zugriffsrechte werden vereinbart.

NONE
Es werden keine Zugriffsrechte vergeben.

***REALM(...)**
Die Realm-Rechte werden vergeben.

NAME = *ALL
Der angegebene Zugriff wird auf alle Realms der Datenbank erlaubt.
Dieser Operand muss für SQL-Anwendungen definiert werden.

NAME = *ALL-EXCEPT(...)
Der angegebene Zugriff wird auf die Realms erlaubt, die hier nicht explizit angegeben werden.

NAME = list-poss(30): <realmname>
Der angegebene Zugriff wird für diese Realms nicht erlaubt.

NAME = list-poss(30): <realmname>
Der angegebene Zugriff wird nur für diese Realms erlaubt.

RIGHT = ALL
Auf die Realms kann sowohl lesend als auch schreibend zugegriffen werden.

RIGHT = RETRIEVAL
Auf die Realms kann nur lesend zugegriffen werden.

***RECORD(...)**
Die Rechte auf Satzarten (in SQL-Anwendungen: Basistabellen) werden vergeben.

NAME = *ALL
Der angegebene Zugriff wird auf alle Satzarten (in SQL-Anwendungen: Basistabellen) der Datenbank erlaubt.

NAME = *ALL-EXCEPT(...)
Der angegebene Zugriff wird auf die Satzarten erlaubt, die hier nicht explizit angegeben werden.

NAME = list-poss(30): <recordname>
Der angegebene Zugriff wird für diese Satzarten nicht erlaubt.

NAME = list-poss(30): <recordname>
Der angegebene Zugriff wird nur für diese Satzarten erlaubt.

RIGHT = ALL
Auf die Satzarten kann sowohl lesend als auch schreibend zugegriffen werden.

RIGHT = RETRIEVAL
Auf die Satzarten kann nur lesend zugegriffen werden.

***SET(...)**
Die Rechte auf Sets (in SQL-Anwendungen: Fremdschlüssel) werden vergeben.

NAME = *ALL
Der angegebene Zugriff wird auf alle Sets (in SQL-Anwendungen: Fremdschlüssel) der Datenbank erlaubt.

NAME = *ALL-EXCEPT(...)
Der angegebene Zugriff wird auf die Sets erlaubt, die hier nicht explizit angegeben werden.

NAME = list-poss(30): <setname>
Der angegebene Zugriff wird für diese Sets nicht erlaubt.

NAME = list-poss(30): <setname>
Der angegebene Zugriff wird nur für diese Sets erlaubt.

RIGHT = ALL
Auf die Sets kann sowohl lesend als auch schreibend zugegriffen werden.

RIGHT = RETRIEVAL
Auf die Sets kann nur lesend zugegriffen werden.

> Bei Anwendungen, die mit BPRIVACY-Gruppenangaben < Version UDS/SQL V1.2 bzw. < UDS-D V1.4 arbeiten, können die Zugriffsrechte durch Angabe des alten Gruppennamens für <host> in dem Operanden *FREE-FORMAT definiert werden. Siehe auch Tabelle 18 ("Rechteprüfung").
>
> BPRIVACY-Gruppenangaben < Version UDS/SQL V1.2, die in ihren Namen ein oder mehrere Leerzeichen enthalten, können über die neue Schnittstelle nicht mehr definiert werden. In diesen Fällen müssen Sie einen neuen Gruppennamen gemäß der SDF-Syntaxregeln vereinbaren.

Beispiel

Es werden folgende Benutzergruppen für die Datenbank VERSAND eingerichtet:

- "D017ZE07_____XXXXX___" : alle Rechte
- "D017ZE07_____YYYYYY__" : Recht auf Wiedergewinnung
- "D017ZE07_____ZZZZZZ__" : ohne Rechte

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=VERSAND.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BPRIVACY
***** START       BPRIVACY     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:40
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:26:40/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:26:40/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.ALL
0YA2: PUBSETS:        *
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
%  UDS0722 UDS AUFTRAG ADD RLOG 150628092640 WIRD AUSGEFUEHRT (ILL1283,11:26:40/0YA2)
%  UDS0356 UDS DURCHFUEHRUNG DER AUFTRAEGE FUER VERSAND  TERMINATED (ILL1309,11:26:40/0YA2)
//ADD-USER-GROUP USER-GROUP-NAME=*FREE-FORMAT (HOST=D017ZE07,USER-ID=XXXXX), -
//  OBJECT=(*REALM(NAME=*ALL,RIGHT=ALL),*RECORD(NAME=*ALL,RIGHT=ALL),*SET(NAME=*ALL,
RIGHT=ALL))
//ADD-USER-GROUP USER-GROUP-NAME=*FREE-FORMAT(HOST=D017ZE07,USER-ID=YYYYYY), -
//  OBJECT=(*REALM(NAME=*ALL,RIGHT=RETRIEVAL),*RECORD(NAME=*ALL,RIGHT=ALL), -
//  *SET(NAME=*ALL,RIGHT=ALL))
//ADD-USER-GROUP USER-GROUP-NAME=*FREE-FORMAT     (HOST=D017ZE07,USER-ID=ZZZZZZ)
//END
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:26:40/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERSAND                  13        111         59         42         20
%  UDS0213 UDS NORMAL BEENDET MIT **************13 DML-STATEMENTS 2017-06-28
(ILLY033,11:26:40/0YA2)

***** DIAGNOSTIC SUMMARY OF BPRIVACY

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END BPRIVACY (UDS/SQL V2.9 0000 ) 2017-06-28 11:26:40
```

4.7.2 END (Kommandoeingabe beenden)

Die Kommandoeingabe wird beendet. Die Ausführungsphase beginnt.

END

Diese Anweisung hat keine Operanden.

4.7.3 GRANT-ACCESS (Zugriffsrechte für eine Benutzergruppe vergeben)

Mit der Anweisung GRANT-ACCESS können Sie Zugriffsrechte an eine Benutzergruppe vergeben auf Realms, Satzarten (RECORDs) und Sets.

Falls eine Benutzergruppe für ein Objekt schon bestimmte Zugriffsrechte besitzt und es werden mit dieser Anweisung dafür Rechte festgelegt, so besitzt die Benutzergruppe diese Rechte zusätzlich.

GRANT-ACCESS

USER-GROUP-NAME = list-poss(6): ***KSET-FORMAT**(...) / ***FREE-FORMAT**(...)

***KSET-FORMAT**(...)

| **HOST** = <host>

| ,**APPLICATION** = <appl>

| ,**KSET** = ***NONE** / list-poss(30): <kset>

***FREE-FORMAT**(...)

| **HOST** = <host>

| ,**USER-ID** = ***NONE** / list-poss(30): <userid>

,**OBJECT** = list-poss(6): ***REALM**(...) / ***RECORD**(...) / ***SET**(...)

***REALM**(...)

| **NAME** = ***ALL** / ***ALL-EXCEPT**(...) / list-poss(30): <realmname>

| | ***ALL-EXCEPT**(...)

| | **NAME** = list-poss(30): <realmname>

| ,**RIGHT** = **ALL** / **RETRIEVAL**

***RECORD**(...)

| **NAME** = ***ALL** / ***ALL-EXCEPT**(...) / list-poss(30): <recordname>

| | ***ALL-EXCEPT**(...)

| | **NAME** = list-poss(30): <recordname>

| ,**RIGHT** = **ALL** /**RETRIEVAL**

***SET**(...)

| **NAME** = ***ALL** / ***ALL-EXCEPT**(...) / list-poss(30): <setname>

| | ***ALL-EXCEPT**(...)

| | **NAME** = list-poss(30): <setname>

| ,**RIGHT** = **ALL** / **RETRIEVAL**

USER-GROUP-NAME = list-poss(6): *KSET-FORMAT(...) / *FREE-FORMAT(...)

Name der Benutzergruppe.

***KSET-FORMAT(...)**
Name der Benutzergruppe.

HOST = <host>
Verarbeitungsrechner der openUTM-Anwendung (siehe *host*, "Benutzergruppen").

APPLICATION = <appl>
Name der openUTM-Anwendung.

KSET = *NONE
Es wird kein KSET-Name angegeben.

KSET = list-poss(6): <kset>
KSET-Name der openUTM-Anwendung.

***FREE-FORMAT(...)**
Name der Benutzergruppe.

HOST = <host>
Verarbeitungsrechner der Anwendung.

USER-ID = *NONE
Es wird keine Kennung angegeben

USER-ID = list-poss(30): <userid>
Kennung der Anwendung.

OBJECT = list-poss(6): *REALM(...) / *RECORD(...) / *SET(...)
Die Zugriffsrechte werden vereinbart.

***REALM (...)**
Die existierenden Realm-Rechte werden verändert.

NAME = *ALL
Der angegebene Zugriff wird auf alle Realms der Datenbank erlaubt. Dieser Operand muss für SQL-Anwendungen definiert werden.

NAME = ALL-EXCEPT (...)
Der angegebene Zugriff wird auf die Realms erlaubt, die hier nicht explizit angegeben werden.

NAME = list-poss(30): <realmname>
Der angegebene Zugriff wird für diese Realms nicht erlaubt.

NAME = list-poss(30): <realmname>
Der angegebene Zugriff wird nur für diese Realms erlaubt.

RIGHT = ALL
Auf die Realms kann sowohl lesend als auch schreibend zugegriffen werden.

RIGHT = RETRIEVAL

Auf die Realms kann nur lesend zugegriffen werden.

***RECORD(...)**
Die existierenden Rechte auf Satzarten (in SQL-Anwendungen: Basistabellen) werden verändert.

NAME = *ALL
Der angegebene Zugriff wird auf alle Satzarten (in SQL-Anwendungen: Basistabellen) der Datenbank erlaubt.

NAME = ALL-EXCEPT (...)
Der angegebene Zugriff wird auf die Satzarten erlaubt, die hier nicht explizit angegeben werden.

NAME = list-poss(30): <recordname>
Der angegebene Zugriff wird für diese Satzarten nicht erlaubt.

NAME = list-poss(30): <recordname>
Der angegebene Zugriff wird nur für diese Satzarten erlaubt.

RIGHT = ALL
Auf die Satzarten kann sowohl lesend als auch schreibend zugegriffen werden.

RIGHT = RETRIEVAL
Auf die Satzarten kann nur lesend zugegriffen werden.

***SET(...)**
Die existierenden Rechte auf Sets (in SQL-Anwendungen: Fremdschlüssel) werden verändert.

NAME = *ALL
Der angegebene Zugriff wird auf alle Sets (in SQL-Anwendungen: Fremdschlüssel) der Datenbank erlaubt.

NAME = ALL-EXCEPT
Der angegebene Zugriff wird auf die Sets erlaubt, die hier nicht explizit angegeben werden.

NAME = list-poss(30): <setname>
Der angegebene Zugriff wird für diese Sets nicht erlaubt.

NAME = list-poss(30): <setname>
Der angegebene Zugriff wird nur für diese Sets erlaubt.

RIGHT = ALL
Auf die Sets kann sowohl lesend als auch schreibend zugegriffen werden.

RIGHT = RETRIEVAL
Auf die Sets kann nur lesend zugegriffen werden.

Beispiel

Es werden der Benutzergruppe "D017ZE07_____ZZZZZZ__" (bisher ohne Rechte) alle Rechte für den Realm AUFTRAGSRLM erteilt.

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=VERSAND.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BPRIVACY
***** START       BPRIVACY     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:40
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:26:40/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:26:40/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.ALL
0YA2: PUBSETS:        *
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
%  UDS0722 UDS AUFTRAG ADD RLOG 150628092640 WIRD AUSGEFUEHRT (ILL1283,11:26:40/0YA2)
%  UDS0356 UDS DURCHFUEHRUNG DER AUFTRAEGE FUER VERSAND  TERMINATED (ILL1309,11:26:41/0YA2)
//GRANT-ACCESS USER-GROUP-NAME=*FREE-FORMAT(HOST=D017ZE07,USER-ID=ZZZZZZ), -
//    OBJECT=(*REALM(NAME=AUFTRAGSRLM,RIGHT=ALL),*RECORD(NAME=*ALL,RIGHT=ALL), -
//    *SET(NAME=*ALL,RIGHT=ALL))
```

```
//SHOW-USER-GROUP USER-GROUP-NAME=*FREE-FORMAT(HOST=D017ZE07,USER-ID=ZZZZZZ),OUTPUT=SYSOUT
//END

DATABASE NAME : $XXXXXXXX.VERSAND
SCHEMA NAME   : ARTIKELVERSAND

*****************************************************************

ACCESS RIGHTS FOR USERGROUP : D017ZE07________ZZZZZZ__


RIGHTS ON REALMS

 +--------------------------------+----------------------+
 !                                !      R I G H T       !
 !                                +-----------+----------+
 ! REALM NAME                     ! RETRIEVAL !  UPDATE   !
 +--------------------------------+-----------+----------+
 ! AUFTRAGSRLM                    !     Y     !     Y     !
 ! BESTELLRLM                     !     N     !     N     !
 ! KLEIDUNG                       !     N     !     N     !
.
.
.


RIGHTS ON RECORDS
 +--------------------------------+----------------------+
 !                                !       R I G H T      !
 !                                +-----------+----------+
 ! RECORD NAME                    ! RETRIEVAL !  UPDATE   !
 +--------------------------------+-----------+----------+
 ! KUNDE                          !     Y     !     Y     !
 ! AUFTRAG                        !     Y     !     Y     !
 ! AUFTR-POS                      !     Y     !     Y     !
.
.
.


RIGHTS ON SETS
 +--------------------------------+----------------------+
 !                                !       R I G H T      !
 !                                +-----------+----------+
 ! SET NAME                       ! RETRIEVAL !  UPDATE   !
 +--------------------------------+-----------+----------+
 ! ERTEILTE-AUFTRAEGE             !     Y     !     Y     !
 ! AUFTR-INHALT                   !     Y     !     Y     !
 ! NOCH-ZU-BEZAHLEN               !     Y     !     Y     !
.
.
.
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:26:41/0YA2)
 0YA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERSAND                   16         82         57         16         16
%  UDS0213 UDS NORMAL BEENDET MIT **************16 DML-STATEMENTS 2017-06-28
(ILLY033,11:26:41/0YA2)
```

```
***** DIAGNOSTIC SUMMARY OF BPRIVACY

            NO WARNINGS
            NO ERRORS
            NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   BPRIVACY     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:41
```

4.7.4 OPEN-DATABASE (Datenbank eröffnen)

Die Anweisung OPEN-DATABASE muss die erste angegebene Anweisung sein.

Mit der Anweisung OPEN-DATABASE legen Sie die Datenbank fest, die mit den nachfolgenden Anweisungen von ONLINE-PRIVACY bzw. von BPRIVACY bearbeitet werden soll.

OPEN-DATABASE
DATABASE-NAME = <dbname>

DATABASE-NAME = <dbname>
Legt die Datenbank fest, bei der die Zugriffsberechtigungen geändert werden sollen. Ein Anwender von ONLINE-PRIVACY bzw. von BPRIVACY kann nur eine Datenbank bearbeiten, die in seiner eigenen Kennung liegt. Eine Datenbank aus einer fremden Kennung kann nur von der TSOS-Kennung des Systemverwalters bearbeitet werden.

> Bei BPRIVACY kann die Anweisung OPEN-DATABASE nicht gegeben werden, wenn die Datenbank über LINK-NAME=DATABASE zugewiesen ist.
> Bei ONLINE-PRIVACY ist die Anweisung OPEN-DATABASE immer erforderlich.
> In diesem Fall wird über LINK-NAME=DATABASE der Konfigurationsname der Independent-DBH-Session angegeben.

4.7.5 REMOVE-USER-GROUP (Benutzergruppen löschen)

Mit der Anweisung REMOVE-USER-GROUP können Sie eine oder mehrere bzw. alle Benutzergruppen mit ihren Rechten löschen.

```
REMOVE-USER-GROUP

USER-GROUP-NAME = ALL / *ALL / *ALL-EXCEPT(...) / list-poss(6): *KSET-FORMAT(...) /
                                *FREE-FORMAT(...)
  *ALL-EXCEPT(...)
    | NAME = list-poss(6): *KSET-FORMAT(...) / *FREE-FORMAT(...)
    |     *KSET-FORMAT(...)
    |         | HOST = <host>
    |         | ,APPLICATION = <appl>
    |         | ,KSET = *NONE / list-poss(30): <kset>
    |     *FREE-FORMAT(...)
    |         | HOST = <host>
    |         | ,USER-ID = *NONE / list-poss(30): <user-id>
  *KSET-FORMAT(...)
    | HOST = <host>
    | ,APPLICATION = <appl>
    | ,KSET = *NONE / list-poss(30): <kset>
  *FREE-FORMAT(...)
    | HOST = <host>
    | ,USER-ID = *NONE / list-poss(30): <userid>
```

USER-GROUP-NAME = ALL / *ALL / *ALL-EXCEPT(...) /
list-poss(6): *KSET-FORMAT(...) / *FREE-FORMAT(...)

Es wird vereinbart, welche Benutzergruppe(n) gelöscht werden soll(en).

ALL / *ALL
Alle vorhandenen Benutzergruppen werden gelöscht.

***ALL-EXCEPT (...)**
Alle Benutzergruppen, die hier nicht explizit angegeben werden, werden gelöscht.

NAME = list-poss(6): *KSET-FORMAT(...) / *FREE-FORMAT(...)
Die hier angegebenen Benutzergruppen werden nicht gelöscht.

***KSET-FORMAT(...)**

HOST = <host>
Verarbeitungsrechner der openUTM-Anwendung (siehe *host*, "Benutzergruppen").

APPLICATION = <appl>
Name der openUTM-Anwendung.

KSET = *NONE
Es wird kein KSET-Name angegeben.

KSET = list-poss(30): <kset>
KSET-Name der openUTM-Anwendung

***FREE-FORMAT(...)**

HOST = <host>
Verarbeitungsrechner der Anwendung

USER-ID = *NONE
Es wird keine Kennung angegeben.

USER-ID = list-poss(30): <userid>
Kennung der Anwendung

***KSET-FORMAT(...)**
Alle hier angegebenen Benutzergruppen werden gelöscht.

HOST = <host>
Verarbeitungsrechner der openUTM-Anwendung (siehe *host*, "Benutzergruppen").

APPLICATION = <appl>
Name der openUTM-Anwendung.

KSET = *NONE
Es wird kein KSET-Name angegeben.

KSET = list-poss(30): <kset>
KSET-Name der openUTM-Anwendung

***FREE-FORMAT(...)**
Alle hier angegebenen Benutzergruppen werden gelöscht.

HOST = <host>
Verarbeitungsrechner der Anwendung

USER-ID = *NONE
Es wird keine Kennung angegeben.

USER-ID = list-poss(30): <userid>
Kennung der Anwendung

Beispiel

Die Benutzergruppe "D017ZE07_____YYYYYY__" wird gelöscht.

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=VERSAND.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BPRIVACY
***** START       BPRIVACY     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:41
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:26:41/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:26:41/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.ALL
0YA2: PUBSETS:        *
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
%  UDS0722 UDS AUFTRAG ADD RLOG 150628092640 WIRD AUSGEFUEHRT (ILL1283,11:26:41/0YA2)
%  UDS0356 UDS DURCHFUEHRUNG DER AUFTRAEGE FUER VERSAND  TERMINATED (ILL1309,11:26:41/0YA2)
//REMOVE-USER-GROUP USER-GROUP-NAME=*FREE-FORMAT    (HOST=D017ZE07,USER-ID=YYYYYY)
//END
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:26:41/0YA2)
 0YA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERSAND                    7         91         60         23         20
%  UDS0213 UDS NORMAL BEENDET MIT ***************7 DML-STATEMENTS 2017-06-28
(ILLY033,11:26:41/0YA2)

***** DIAGNOSTIC SUMMARY OF BPRIVACY

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   BPRIVACY     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:41
```

4.7.6 REVOKE-ACCESS (Einer Benutzergruppe Zugriffsrechte entziehen)

Mit der Anweisung REVOKE-ACCESS können Sie einer Benutzergruppe die Zugriffsrechte auf Realms, Satzarten (RECORDs) und Sets entziehen.

Es werden nur die hier angegebenen Zugriffsrechte entzogen, d.h. eventuell weitere (hier nicht aufgezählte) Zugriffsrechte bleiben erhalten.

REVOKE-ACCESS

USER-GROUP-NAME = list-poss(6): ***KSET-FORMAT**(...) / ***FREE-FORMAT**(...)

***KSET-FORMAT**(...)

| **HOST** = <host>

| ,**APPLICATION** = <appl>

| ,**KSET** = **<u>*NONE</u>** / list-poss(30): <kset>

***FREE-FORMAT**(...)

| **HOST** = <host>

| ,**USER-ID** = **<u>*NONE</u>** / list-poss(30): <userid>

,**OBJECT** = list-poss(6): ***REALM**(...) / ***RECORD**(...) / ***SET**(...)

***REALM**(...)

| **NAME** = **<u>*ALL</u>** / ***ALL-EXCEPT**(...) / list-poss(30): <realmname>

| | ***ALL-EXCEPT**(...)

| | **NAME** = list-poss(30): <realmname>

| ,**RIGHT** = **ALL** / **UPDATE** / **RETRIEVAL**

***RECORD**(...)

| **NAME** = **<u>*ALL</u>** / ***ALL-EXCEPT**(...) / list-poss(30): <recordname>

| | ***ALL-EXCEPT**(...)

| | **NAME** = list-poss(30): <recordname>

| ,**RIGHT** = **ALL** / **UPDATE** / **RETRIEVAL**

***SET**(...)

| **NAME** = **<u>*ALL</u>** / ***ALL-EXCEPT**(...) / list-poss(30): <setname>

| | ***ALL-EXCEPT**(...)

| | **NAME** = list-poss(30): <setname>

| ,**RIGHT** = **ALL** / **UPDATE** / **RETRIEVAL**

USER-GROUP-NAME = list-poss(6): *KSET-FORMAT(...) / *FREE-FORMAT(...)
Name der Benutzergruppe.

***KSET-FORMAT(...)**
Name der Benutzergruppe.

HOST = <host>
Verarbeitungsrechner der openUTM-Anwendung (siehe *host*, "Benutzergruppen").

APPLICATION = <appl>
Name der openUTM-Anwendung.

KSET = *NONE
Es wird kein KSET-Name angegeben.

KSET = list-poss(30): <kset>
KSET-Name der openUTM-Anwendung.

***FREE-FORMAT(...)**
Name der Benutzergruppe.

HOST = <host>
Verarbeitungsrechner der Anwendung.

USER-ID = *NONE
Es wird keine Kennung angegeben.

USER-ID = list-poss(30): <userid>
Kennung der Anwendung.

OBJECT = list-poss(6): *REALM(...) / *RECORD(...) / *SET(...)
Die Zugriffsrechte werden vereinbart.

***REALM(...)**
Die existierenden Realm-Rechte werden verändert.

NAME = *ALL
Der angegebene Zugriff wird auf alle Realms der Datenbank verboten.

NAME = *ALL-EXCEPT (...)
Der angegebene Zugriff wird auf die Realms verboten, die hier nicht explizit angegeben werden.

NAME = list-poss(30): <realmname>
Der angegebene Zugriff wird für diese Realms nicht verboten.

NAME = list-poss(30): <realmname>
Der angegebene Zugriff wird nur für diese Realms verboten.

RIGHT = ALL
Auf die Realms kann nicht mehr zugegriffen werden.

RIGHT = UPDATE
Auf die Realms kann nicht mehr schreibend zugegriffen werden.

RIGHT = RETRIEVAL
Diese Angabe ist nur möglich, wenn die angegebene Benutzergruppe kein Recht zum Ändern (UPDATE) hat.
Auf die Realms kann nicht mehr zugegriffen werden.

***RECORD(...)**
Die existierenden Rechte auf Satzarten (in SQL-Anwendungen: Basistabellen) werden verändert.

NAME = *ALL
Der angegebene Zugriff wird auf alle Satzarten (in SQL-Anwendungen: Basistabellen) der Datenbank verboten.

NAME = *ALL-EXCEPT (...)
Der angegebene Zugriff wird auf die Satzarten verboten, die hier nicht explizit angegeben werden.

NAME = list-poss(30): <recordname>
Der angegebene Zugriff wird für diese Satzarten nicht verboten.

NAME = list-poss(30): <recordname>
Der angegebene Zugriff wird nur für diese Satzarten verboten.

RIGHT = ALL
Auf die Satzarten kann nicht mehr zugegriffen werden.

RIGHT = UPDATE
Auf die Satzarten kann nicht mehr schreibend zugegriffen werden.

RIGHT = RETRIEVAL
Diese Angabe ist nur möglich, wenn die angegebene Benutzergruppe kein Recht zum Ändern (UPDATE) hat.
Auf die Satzarten kann nicht mehr zugegriffen werden.

***SET(...)**
Die existierenden Rechte auf Sets (in SQL-Anwendungen: Fremdschlüssel) werden verändert.

NAME = *ALL
Der angegebene Zugriff wird auf alle Sets (in SQL-Anwendungen: Fremdschlüssel) der Datenbank verboten.

NAME = *ALL-EXCEPT(...)
Der angegebene Zugriff wird auf die Sets verboten, die hier nicht explizit angegeben werden.

NAME = list-poss(30): <setname>
Der angegebene Zugriff wird für diese Sets nicht verboten.

NAME = list-poss(30): <setname>
Der angegebene Zugriff wird nur für diese Sets verboten.

RIGHT = ALL
Auf die Sets kann nicht mehr zugegriffen werden.

RIGHT = UPDATE
Auf die Sets kann nicht mehr schreibend zugegriffen werden.

RIGHT = RETRIEVAL
Diese Angabe ist nur möglich, wenn die angegebene Benutzergruppe kein Recht zum Ändern (UPDATE) hat.
Auf die Sets kann nicht mehr zugegriffen werden.

Beispiel

Der Benutzergruppe "D017ZE07_____ZZZZZZ__" wird das Update-Recht auf den Realm AUFTRAGSRLM entzogen.

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=VERSAND.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BPRIVACY
***** START        BPRIVACY     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:41
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:26:41/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:26:41/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.ALL
0YA2: PUBSETS:        *
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
%  UDS0722 UDS AUFTRAG ADD RLOG 150628092640 WIRD AUSGEFUEHRT (ILL1283,11:26:41/0YA2)
%  UDS0356 UDS DURCHFUEHRUNG DER AUFTRAEGE FUER VERSAND  TERMINATED (ILL1309,11:26:41/0YA2)
//REVOKE-ACCESS USER-GROUP-NAME=*FREE-FORMAT(HOST=D017ZE07,USER-ID=ZZZZZZ), -
//    OBJECT=(*REALM(NAME=AUFTRAGSRLM,RIGHT=UPDATE))
//SHOW-USER-GROUP USER-GROUP-NAME=ALL,OUTPUT=SYSOUT
//END

DATABASE NAME : $XXXXXXXX.VERSAND
SCHEMA NAME   : ARTIKELVERSAND

*********************************************************************

ACCESS RIGHTS FOR USERGROUP : D017ZE07_____ZZZZZZ__


RIGHTS ON REALMS

 +-------------------------------+----------------------+
 !                               !      R I G H T       !
 !                               +----------+-----------+
 ! REALM NAME                    ! RETRIEVAL !  UPDATE   !
 +-------------------------------+----------+-----------+
 ! AUFTRAGSRLM                   !     Y     !     N     !
 ! BESTELLRLM                    !     Y     !     Y     !
 ! KLEIDUNG                      !     Y     !     Y     !
.
.
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:26:41/0YA2)
 0YA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERSAND                   17         84         58         16         16
%  UDS0213 UDS NORMAL BEENDET MIT **************17 DML-STATEMENTS 2017-06-28
(ILLY033,11:26:41/0YA2)

 ***** DIAGNOSTIC SUMMARY OF BPRIVACY

              NO WARNINGS
              NO ERRORS
              NO SYSTEM-ERRORS

 ***** END OF DIAGNOSTIC SUMMARY
 ***** NORMAL END   BPRIVACY     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:41
```

4.7.7 SHOW-USER-GROUP (Informationen über Benutzergruppen ausgeben)

Es werden Informationen über Benutzergruppen ausgegeben, d.h. über die vergebenen Zugriffsrechte auf die Datenbankobjekte für die Benutzergruppen. Die Realms, Satzarten und Sets werden nach Referenznummern aufsteigend ausgegeben.

```
SHOW-USER-GROUP

USER-GROUP-NAME = ALL / *ALL / *ALL-EXCEPT(...) / list-poss(6): *KSET-FORMAT(...) /
                                                 *FREE-FORMAT(...)
  *ALL-EXCEPT(...)
    | NAME = list-poss(6): *KSET-FORMAT(...) / *FREE-FORMAT(...)
    |     *KSET-FORMAT(...)
    |         | HOST = <host>
    |         | ,APPLICATION = <appl>
    |         | ,KSET = *NONE / list-poss(30): <kset>
    |     *FREE-FORMAT(...)
    |         | HOST = <host>
    |         | ,USER-ID = *NONE / list-poss(30): <userid>
  *KSET-FORMAT(...)
    | HOST = <host>
    | ,APPLICATION = <appl>
    | ,KSET = *NONE / list-poss(30): <kset>
  *FREE-FORMAT(...)
    | HOST = <host>
    | ,USER-ID = *NONE / list-poss(30): <userid>
,OUTPUT = list-poss: SYSLST / SYSOUT
```

USER-GROUP-NAME = ALL / *ALL / *ALL-EXCEPT(...) / list-poss(6): *KSET-FORMAT(...) /*FREE-FORMAT(...)
Die Rechte werden ausgegeben.

ALL / *ALL
Die Rechte von allen Benutzergruppen werden ausgegeben.

***ALL-EXCEPT (...)**
Es werden die Rechte der Benutzergruppen ausgegeben, die hier nicht explizit angegeben sind.

NAME = list-poss(6): *KSET-FORMAT(...) / *FREE-FORMAT(...)
Die Rechte der angegebenen Benutzergruppen werden nicht ausgegeben.

***KSET-FORMAT(...)**

HOST = <host>
Verarbeitungsrechner der openUTM-Anwendung (siehe *host*, "Benutzergruppen").

APPLICATION = <appl>
Name der openUTM-Anwendung

KSET = *NONE
Es wird kein KSET-Name angegeben.

KSET = list-poss (30): <kset>
KSET-Name der openUTM-Anwendung

***FREE-FORMAT(...)**

HOST = <host>
Verarbeitungsrechner der Anwendung

USER-ID = *NONE

USER-ID = list-poss(30): <userid>
Kennung der Anwendung

***KSET-FORMAT(...)**
Die Rechte der angegebenen Benutzergruppen werden ausgegeben.

HOST = <host>
Verarbeitungsrechner der openUTM-Anwendung (siehe *host*, "Benutzergruppen").

APPLICATION = <appl>
Name der openUTM-Anwendung

KSET = *NONE
Es wird kein KSET-Name angegeben.

KSET = list-poss(30): <kset>
KSET-Name der openUTM-Anwendung

***FREE-FORMAT(...)**

HOST = <host>
Verarbeitungsrechner der Anwendung

USER-ID = *NONE
Es wird keine Kennung angegeben.

USER-ID = list-poss(30): <userid>
Kennung der Anwendung

OUTPUT = list-poss: SYSLIST / SYSOUT
Die Informationen werden ausgegeben.

SYSLST
Die Ausgabe erfolgt auf SYSLST.

SYSOUT
Die Ausgabe erfolgt auf SYSOUT.

Beispiel

siehe Beispiel zu GRANT-ACCESS

4.7.8 UNDO (Anweisung rückgängig machen)

Die zuletzt eingegebene, korrekte Anweisung (außer UNDO selbst) wird nicht ausgeführt. Mit einer weiteren Anweisung UNDO wird die vorletzte Anweisung (außer UNDO) nicht ausgeführt usw.

UNDO

Diese Anweisung hat keine Operanden.

4.8 Kommandofolge zum Starten von ONLINE-PRIVACY

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version

02 /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=konfigurationsname

03 /START-UDS-ONLINE-PRIVACY

04 OPEN-DATABASE DATABASE-NAME=dbname

05 übrige online-privacy-anweisungen

06 END
```

Erläuterung

01 Geben Sie mit dem Kommando SELECT-PRODUCT-VERSION an, welche UDS/SQL-Version genutzt werden soll, da mit IMON im Software Configuration Inventory (SCI) mehrere UDS/SQL-Versionen parallel installiert und mehrere Versionen des UDS/SQL-Subsystems vorgeladen sein können.

02 Mit dem Kommando SET-FILE-LINK weisen Sie den Konfigurationsnamen FILE-NAME= *konfigurationsname* über den Linknamen DATABASE zu.
Die UDS/SQL-Konfiguration, mit der ONLINE-PRIVACY zusammenarbeiten soll und an die die zu bearbeitende Datenbank angeschlossen ist, müssen Sie mit diesem Kommando bekannt machen.

03 ONLINE-PRIVACY muss in der Kennung ablaufen, in der die zu bearbeitende Datenbank steht. Ist dies nicht der Fall, so wird der Zugriff auf die Datenbank vom DBH mit dem Statuscode 901 abgewiesen.

04 Mit der Anweisung OPEN-DATABASE DATABASE-NAME=... legen Sie die zu bearbeitende Datenbank fest. Im Unterschied zum Dienstprogramm BPRIVACY ist diese Anweisung bei ONLINE-PRIVACY zwingend erforderlich.
Mit jedem ONLINE-PRIVACY-Lauf können Sie nur eine einzige Datenbank bearbeiten.
Ein Zugriff auf eine Datenbank in einer Remote Konfiguration über UDS-D ist mit ONLINE-PRIVACY nicht möglich.

4.9 Kommandofolge zum Starten von BPRIVACY

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
02  [/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR]
03  /START-UDS-BPRIVACY
04  bprivacy-anweisungen
05  END
```

Erläuterung

01 Versionsabhängiges Modul des linked-in DBH der entsprechenden Version wird nachgeladen (siehe Handbuch „Anwendungen programmieren“, Abschnitt „UDS/SQL-TIAM-Anwenderprogramme binden, laden und starten“).

02 Wenn Sie die Datenbank über LINK-NAME=DATABASE zuweisen, dürfen Sie die BPRIVACY-Anweisung OPEN-DATABASE nicht angeben.
Wenn Sie die Datenbank nicht über LINK-NAME=DATABASE zuweisen, müssen Sie die BPRIVACY-Anweisung OPEN-DATABASE angeben.

03 BPRIVACY muss in der Kennung ablaufen, in der die zu bearbeitende Datenbank steht. Das UDS/SQL-Dienstprogramm kann auch mit den Aliasnamen BPRIVACY und START-UDS-AUTHORIZATION gestartet werden.

5 Daten speichern und entladen (BINILOAD, BOUTLOAD)

In diesem Kapitel sind die beiden Dienstprogramme BINILOAD und BOUTLOAD beschrieben, mit denen Sie Daten speichern bzw. entladen können.

Das Dienstprogramm BINILOAD können Sie jederzeit ablaufen lassen, um Daten in eine leere oder teilweise geladene Datenbank zu speichern.

Das Dienstprogramm BOUTLOAD verwenden Sie, um ganze Satzarten aus der Datenbank zu kopieren, zu löschen oder zu entladen, z.B. bei Umstrukturierungen.

Beide Dienstprogramme berücksichtigen beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb“, Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

5.1 Sätze in die Datenbank speichern mit BINILOAD

Ob Sie mit Hilfe eines Anwenderprogramms oder mit BINILOAD Daten in die Datenbank speichern, diese Entscheidung soll Ihnen die folgende Gegenüberstellung erleichtern.

Stichwort	Anwenderprogramm	BINILOAD
Einsatz vorwiegend	um die Datenbank auf den neuesten Stand zu bringen.	um eine Datenbank zu laden.
Speichert	jeden Satz einer Satzart einzeln in die Datenbank ein.	große Mengen von Sätzen einer Satzart in einem oder mehreren Läufen in die Datenbank ein.
Geeignet zum Laden	einer Datenbank, die viele Set-Occurrences mit wenigen Membersätzen hat.	einer Datenbank, deren Set-Occurrences große Mengen von Membersätzen haben.
Beim Einfügen in Sets	muss für jeden Membersatz die richtige Set-Occurrence und die Position innerhalb dieser Set-Occurrence einzeln bestimmt werden.	werden alle Membersätze, die zu einer bestimmten Set-Occurrence gehören sollen, als Folge von Sätzen einer Eingabedatei bereitgestellt und gemeinsam verarbeitet.
Die Effizienz beim Laden der Datenbank	ist unabhängig von der Anzahl der Sätze, die gespeichert werden.	erhöht sich mit der Anzahl der Sätze, die gespeichert werden.

Tabelle 20: Gegenüberstellung von Anwenderprogramm und BINILOAD

BINILOAD ist für das Laden der Datenbank konzipiert und kann eine große Anzahl von Sätzen einer Satzart effizient laden.
Ob und wie viel Verarbeitungszeit BINILOAD gegenüber einem Anwenderprogramm spart, richtet sich nach der Anzahl der Sätze, die gespeichert werden, und der Struktur der Datenbank.

> ℹ Da BINILOAD nicht mit Before-Images arbeitet, ist es sinnvoll, DBDIR und mindestens die beim Laden benötigten Realms vor dem Ablauf zu sichern; dies gilt besonders dann, wenn Sie auf den Prüflauf verzichten wollen.

Mit BINILOAD können Sie Sätze einer Satzart in eine leere oder teilweise leere Datenbank speichern. Grundsätzlich stellen Sie alle zu speichernden Sätze in einer Eingabedatei für BINILOAD bereit. Alle Sätze, die während eines Laufs geladen werden, speichert BINILOAD in denselben Realm ein. Sätze derselben Satzart können schon in der Datenbank enthalten sein. Pro Lauf bearbeitet BINILOAD nur Sätze einer Satzart.

BINILOAD kann

- Daten aus CSV-Dateien laden;
- die Sätze der Eingabedatei vorsortieren;
- Sätze mit Feldern fester und variabler Länge speichern;

- beim Speichern von Membersätzen entweder alle Sätze der Eingabedatei in eine von Ihnen bestimmte Set-Occurrence einhängen, oder die einzelnen Set-Occurrences, in die die Sätze einzuhängen sind, mit Hilfe des Schlüsselwertes des jeweiligen Ownersatzes auswählen.

Die Membersatzfolge legt BINILOAD entsprechend den jeweils vorgegebenen Sortierkriterien fest und baut die Set-Occurrence-Tabellen so auf, dass nur wenige Zugriffe auf die Blöcke der Datenbank nötig sind.

Sie können aber auch die gewünschte Membersatzfolge durch eine Zeichenfolge in den Sätzen der Eingabedatei selbst festlegen (nur bei Sets mit ORDER ungleich SORTED oder SORTED INDEXED, z.B. ORDER IS LAST).

BINILOAD arbeitet nicht mit dem DBH, sondern greift selbst auf die Datenbank zu. Die Datenbanksätze (einschließlich Set Connection Data) und Tabellen (Adresslisten, Listen, SEARCH-Key-Tabellen usw.) erzeugt BINILOAD in Arbeitsdateien bzw. im Hauptspeicher und stellt die Datenbankseiten vollständig zusammen, bevor BINILOAD sie in die Datenbank überträgt.

5.1.1 Beschreibung der Funktionen

In Sets einfügen

BINILOAD kann Sätze, die in die Datenbank gespeichert werden, in Sets einfügen, wobei zwischen Owner- und Membersatzarten zu unterscheiden ist:

- Ownersatzart:
  BINILOAD erzeugt z.B. in einem CHAIN-Set für jeden gespeicherten Ownersatz eine leere Set-Occurrence;
- Membersatzart eines oder mehrerer Sets:
  Sie müssen mit der INSERT-Anweisung (siehe Abschnitt „Anweisungen für BINILOAD") festlegen, in welche Sets BINILOAD die Sätze einfügen soll.

  Dies gilt auch für Sets, in denen die Satzart als AUTOMATIC-Member definiert ist.

Set-Occurrence auswählen

Soll BINILOAD Membersätze in einen Set einfügen, so benötigt BINILOAD Informationen zum Auswählen der Set-Occurrence, in die die Membersätze einzufügen sind. Dies geschieht, indem BINILOAD den Ownersatz auswählt. BINILOAD kann den Ownersatz über folgende Schlüsselwerte auswählen:

- Wert des CALC-Key (siehe Handbuch „Entwerfen und Definieren“, LOCATION MODE-Klausel)
- Wert des SEARCH-Key; auf Satzartebene definiert entweder als INDEX-SEARCH-Key oder als CALC-SEARCH-Key (siehe Handbuch „Entwerfen und Definieren“, SEARCH-KEY-Klausel)
- Wert des Database Key; diese Methode ist immer möglich, da der Wert des Database Key ein eindeutiges Kennzeichen innerhalb der Datenbank ist

Wenn der Ownersatz Member in einem SYSTEM-Set ist, kann BINILOAD ihn über folgende Schlüsselwerte auswählen:

- Wert des ASC-/DESC-Key; bei einem sortierten SYSTEM-Set
- Wert des SEARCH-Key; auf Set-Ebene als INDEX-SEARCH-Key oder als CALC-SEARCH-Key definiert (siehe Handbuch „Entwerfen und Definieren“, Direktzugriff).

Den Schlüssel des Ownersatzes, über den die Set-Occurrence auszuwählen ist, können Sie BINILOAD auf zwei Arten mitteilen:

- Sie geben die Position des Schlüssels in den Sätzen der Eingabedatei an. In diesem Fall kann der Wert des Schlüssels von Satz zu Satz variieren. Demzufolge hängt BINILOAD die Sätze mit verschiedenen Schlüsselwerten in verschiedene Set-Occurrences ein.
- Sie geben den Schlüssel als Literal in der OWNER-Anweisung von BINILOAD an. In diesem Fall gilt der Schlüsselwert für alle zu speichernden Sätze. BINILOAD hängt also alle Sätze der Eingabedatei in **eine** Set-Occurrence ein.

Folgende Punkte müssen Sie beim Auswählen einer Set-Occurrence berücksichtigen:

- Duplikate
  Wählen Sie die Set-Occurrence aus durch Angabe des Schlüsselwertes (CALC-Key, ASC-/DESC-Key, SEARCH-Key) der Ownersatzart und existieren mehrere Sätze mit diesem Schlüsselwert (DUPLICATES ARE ALLOWED), dann wählt BINILOAD aus den Ownersätzen mit gleichem Schlüsselwert einen aus, ohne dass der Anwender vorhersehen kann, welcher dies ist.
- MANUAL-Member
  Sollen die Sätze der Eingabedatei in einen Set eingefügt werden, in dem ihre Satzart als MANUAL-Member

definiert ist, so müssen Sie das Einhängen bzw. das Nichteinhängen angeben mit Hilfe der INSERT-Anweisung und durch Kennzeichnen der Eingabesätze bei wahlweisem Einhängen in einem BINILOAD-Lauf.

- Membersatzfolge
  Ist der Set, in den BINILOAD die zu speichernden Sätze als Membersätze einhängen soll, mit ORDER IS FIRST /LAST/NEXT/PRIOR oder IMMATERIAL definiert, dann können Sie mit der SET ORDER-Anweisung von BINILOAD angeben, wie BINILOAD die Sätze in die Set-Occurrences einhängen soll:
  - aufsteigend sortiert, nach dem Inhalt eines Feldes der Eingabesätze
  - in der Reihenfolge, in der sie in der Eingabedatei vorliegen

  In den anderen Fällen gilt die Reihenfolge, die in der ORDER-Klausel der Set-Beschreibung definiert wurde.
- Set-Occurrence nicht leer
  Sollen die zu speichernden Sätze als Membersätze in eine Set-Occurrence eingehängt werden, in der bereits Membersätze eingetragen sind, muss die Struktur der vorhandenen Set-Occurrence berücksichtigt werden.

  Diese Struktur wird bestimmt durch folgende Klauseln:
  - MODE-Klausel (SSL) mit POINTER-ARRAY, LIST, CHAIN
  - ORDER-Klausel (DDL) mit FIRST, LAST, NEXT, PRIOR, IMMATERIAL, SORTED (nur möglich bei CHAIN), SORTED INDEXED.

  DDL und SSL siehe Handbuch „Entwerfen und Definieren“.

  Folgendes müssen Sie dabei beachten:
  - Bei MODE IS CHAIN mit SORTED, SORTED INDEXED kann eine ungünstige Verkettungsstruktur entstehen.
  - Bei MODE IS LIST kann BINILOAD keine zusätzlichen Sätze in eine bereits vorhandene Set-Occurrence einfügen. Dies gilt auch für verteilbare Listen.
  - Bei MODE IS LIST benötigt BINILOAD für Stufe 0 jeder Set-Occurrence mindestens eine Seite.

In die Datenbank speichern

BINILOAD ist dafür gedacht, größere Mengen von Sätzen in die Datenbank zu speichern. Da BINILOAD keine teilweise gefüllten Seiten verwendet, kann mehr Speicherplatz verbraucht werden, als es beim Laden mit einem Anwenderprogramm der Fall wäre.

Es kann vorkommen, dass Eingabedaten, BINILOAD-Anweisungen und Angaben bei Schema-DDL und SSL unvereinbar sind. Daher bietet BINILOAD die Möglichkeit, die Daten der Eingabedatei zu prüfen bevor sie endgültig gespeichert werden.

Durch die Angabe EXECUTION WITH CHECK wird diese erste Phase beim Einspeichern mit BINILOAD, die Tabellenaufbauphase, einmal durchlaufen, ohne Änderungen in der Datenbank vorzunehmen. Dabei werden auftretende Fehler erkannt und entsprechende Meldungen ausgegeben.

Sind keine Fehler aufgetreten, so setzt BINILOAD wieder am Anfang des Laufs auf und die Eingabedaten werden in die Datenbank gespeichert.

Durch die Angabe EXECUTION WITHOUT CHECK wird sofort in die Datenbank geschrieben. Tritt ein **Fehler** auf, wird der BINILOAD-Lauf abgebrochen, Fehlermeldungen werden ausgegeben und die **Datenbank ist inkonsistent**.

- Abhängigkeiten zwischen Schema-DDL und BINILOAD-Anweisungen:
  - DUPLICATES ARE NOT ALLOWED und Duplikate vorhanden

Wenn trotz Angabe von DUPLICATES ARE NOT ALLOWED
Sätze in der Eingabedatei enthalten sind, die den gleichen Schlüsselwert haben, dann erhalten Sie die Meldung;

```
DUPLICATE KEYS OR DBKEYS FOUND / REC REF‘S OR RSQ‘S OUT OF RANGE
SEE PRINTER OUTPUT
```

Zusätzlich werden im BINILOAD-Ablaufprotokoll maximal die ersten 60 Bytes des Schlüssels in Zeichendarstellung und maximal die ersten 30 Bytes des Schlüssels in sedezimaler Darstellung ausgegeben. Die Meldung wird für jeden mehrfach festgestellten Schlüsselwert wiederholt. BINILOAD bricht nach Prüfung aller Eingabesätze mit der Meldung ab:

```
ABNORMAL END BINILOAD
```

BINILOAD kann nur Duplikate innerhalb der Eingabedatei feststellen. Duplikate zwischen Eingabesätzen und Datenbanksätzen werden nicht erkannt.

- WITHIN-Klausel und RECORD AREA-Anweisung von BINILOAD

  Wenn in der WITHIN-Klausel der DDL mehr als ein Realm definiert wurde, dann müssen Sie den Realm, in den die Sätze zu speichern sind, in der RECORD AREA-Anweisung von BINILOAD angeben.

- Abhängigkeiten zwischen SSL- und BINILOAD-Anweisungen:
  - DATABASE-KEY-TRANSLATION-TABLE-Klausel

    Diese Klausel legt die Größe und Lage der DBTT für eine Satzart fest und bestimmt damit die Anzahl der Sätze, die höchstens in der Datenbank gespeichert werden können. Die Eingabedatei darf daher nicht mehr Sätze enthalten, als freie Einträge in der DBTT der betreffenden Satzart vorhanden sind.
  - PLACEMENT OPTIMIZATION-Klausel

    Bei Verwendung dieser Klausel werden die Membersätze zu diesem Set, entsprechend dem Sortierkriterium, dicht in aufeinander folgende Seiten gepackt; der Member wird nicht in die Seite des Owner gespeichert.
  - SSL-Angaben mit ATTACHED

    MODE IS POINTER-ARRAY ATTACHED

    MODE IS LIST ATTACHED

    INDEX NAME IS *indexname* PLACING IS ATTACHED

    Die Angabe von ATTACHED in der SSL ist unwirksam.BINILOAD speichert Adresslisten, Listen und Indizes aller Stufen in leere Datenbankseiten; diese Informationselemente werden immer DETACHED innerhalb des Realm des Owner gespeichert.
  - SSL-Angaben mit DETACHED

    Die Angabe DETACHED WITHIN *realmname* in der SSL bewirkt, dass die Adresstabellen und Indizes in dem angegebenen Realm gespeichert werden. Einzige Ausnahme bilden die verteilbaren Listen. Hier wird der Tabellenteil (Stufen >0) und ein möglicher indirekter Hashbereich in *realmname* gespeichert.
  - POPULATION-Klausel im Set-Eintrag der SSL und FILLING-Anweisung von BINILOAD

    In der FILLING-Anweisung kann der Füllgrad für Tabellenseiten angegeben werden. Auf diese Weise ist es möglich, ohne eine unmittelbare Erweiterung und Reorganisation für diese Informationselemente zusätzliche Membersätze durch den Database Handler zu laden. Der für diese Tabellenseiten zu reservierende Speicherplatz wird ausschließlich durch den Füllgrad bestimmt und nicht durch die Set-POPULATION-Klausel in der SSL.

Schema-DDL und SSL siehe Handbuch „Entwerfen und Definieren“.

5.1.2 Eingabedatei bereitstellen und BINILOAD-Lauf vorbereiten

Die Sätze, die Sie mit dem Dienstprogramm BINILOAD speichern wollen, müssen Sie in einer Eingabedatei bereitstellen.

Die Sätze der Eingabedatei müssen alle den gleichen Aufbau haben. Die gleiche Eingabedatei kann aber zum Speichern von Datenbanksätzen mit unterschiedlichem Aufbau benutzt werden, wenn unterschiedliche Felder des Eingabesatzes ausgewählt werden.
Ein Satz der Eingabedatei kann, außer Feldern für den Datenbanksatz, Felder mit folgenden Informationen enthalten:

- Benutzerinformation; wird von BINILOAD nicht ausgewertet
- Schlüsselwerte, die BINILOAD auswertet, um die richtige Set-Occurrence zu bestimmen
- Informationen, die BINILOAD auswertet, um die Reihenfolge der Membersätze zu entscheiden
- Informationen, die BINILOAD auswertet, um das Einhängen bzw. Nicht-Einhängen von Membersätzen zu kennzeichen.

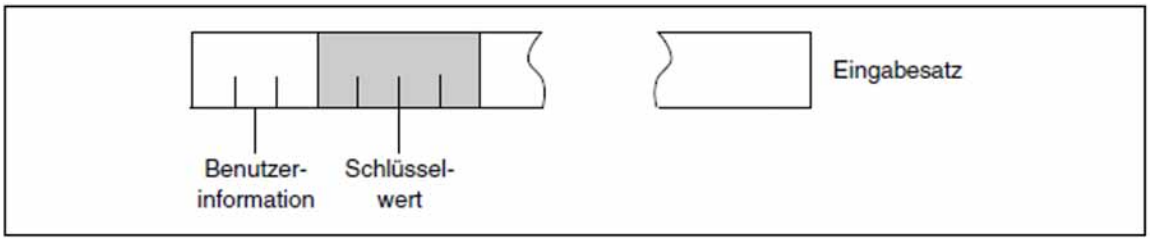

Bild 18: Beispiel zum Aufbau eines Satzes der Eingabedatei

Mit Hilfe von BINILOAD-Anweisungen geben Sie die Position und die Länge der Felder an, die in den Datenbanksatz zu übertragen sind (siehe Tabelle 23, RECORD DISPL-Anweisung, "Anweisungen für BINILOAD").

Die Feldinhalte werden in dem Format in die Datenbank gespeichert, in dem sie in den Sätzen der Eingabedatei vorkommen. Das bedeutet, dass die angegebenen Feldinhalte nicht in den Typ konvertiert werden, der in der Schema-DDL für diese Felder angegeben wurde (siehe Handbuch „Entwerfen und Definieren“, Definieren eines alphanumerischen Feldes fester Länge).

Die Eingabedatei darf auf Platte oder Band stehen. In beiden Fällen kann die Eingabedatei eine SAM-oder ISAM-Datei (EDT-Format) mit Sätzen gleicher Länge im festen oder variablen Satzformat (RECFORM=V oder F) sein (siehe Tabelle 22, USER RECORD LENGTH-Anweisung, "Anweisungen für BINILOAD").

Erfolgt die Eingabe aus einer nicht katalogisierten Banddatei, so muss im /CREATE-FILE-Kommando für die Eingabedatei auch STATE=FOREIGN angegeben werden.

Den Namen der Eingabedatei geben Sie in der INPUT-FILE-Anweisung an, ihre Satzlänge mit der Anweisung USER RECORD LENGTH.

Nach dem Beispiel für die BINILOAD-Kommandofolge finden Sie am Ende dieses Kapitels ein Beispiel für eine BINILOAD-Eingabedatei.

5.1.3 Eingabedatei im CSV-Format lesen und BINILOAD-Lauf vorbereiten

BINILOAD kann Daten aus CSV-Dateien lesen, die von BOUTLOAD oder anderen Tools nach den Regeln für das CSV-Dateiformat erstellt wurden.

Eine von BOUTLOAD erzeugte Eingabedatei im CSV-Format kann Zeilen mit den folgenden Informationen enthalten:

```
(1)     BOUTLOAD;CSV V1.20;2018-01-16;07:45:27;

(2)     DBNAME;DATABASE NAME;BINICSV;

(3)     INFO01;RECORD NAME;REC-001;

(4)     INFO02;RECORD REF;2;

(5)     INFO03;REALM NAME;AREA1

(6)     INFO04;REALM REF;3

(7)     FIELDS;DB Key Ref;DB Key RSQ;Member SYS-1;Owner DB Key Ref S1;Owner DB Key
RSQ S1;R1;R2;

(8)     RECORD;2;1;Y;3;2;”AAA”;1;
```

Zeile (1) kann den Namen des BOUTLOAD Dienstprogramms, die zugehörige Ausgabeformat-Version des Dienstprogramms und Datum und Uhrzeit der Erstellung der CSV-Ausgabe enthalten.

Die nächsten Zeilen (2) – (6) enthalten den Datenbanknamen, den Satznamen und die Satzart. Wenn der Realm-Name angegeben wurde, dann enthält die Kopfzeile auch den Realm-Namen und die Realm-Nummer.

Die Zeilen (1)-(6) sind in der CSV-Datei optional und werden während des Ladens mit BINILOAD ignoriert.

Die Kopfzeile (7) und die Inhaltszeile (8) sind verbindlich, die Kontrollfelder mit den Namen FIELDS und RECORD sind optional.

Die Feldnamen in der Kopfzeile sollten den folgenden Namenskonventionen entsprechen:

- “DB Key Ref” und “DB Key RSQ” – Feldnamen für den Datenbankschlüssel des Satzes
- “Member *set-name*” – Feldname für ein ein Byte langes Feld mit dem Inhalt.

Y = Member, der in den SYSTEM-Set set-name eingefügt wurde.

N = Member, der nicht in den SYSTEM-Set set-name eingefügt wurde.

(für alle singulären Sets, in denen der Satz ein Member ist, außer für Member vom Typ MANDATORY AUTOMATIC)

- “Owner DB Key Ref *set-name*” und “Owner DB Key RSQ *set-name*” - Feldnamen für den Datenbankschlüssel des Owners in *set-name*.
- Die Feldnamen entsprechend des Benutzerschemas;
- Area ref - im Falle eines Satztyps, der an die Realms verteilt wird. wenn seine Sätze von multiplen Realms kopiert werden.

Die einzelnen Werte werden durch Semikolon (";") getrennt.

Die Reihenfolge der Werte in der Inhaltszeile (8) muss mit der Reihenfolge der Felder in der Kopfzeile (7) übereinstimmen.

Die Felder in der CSV-Datei können in beliebiger Reihenfolge angeordnet sein.

Die korrekte Reihenfolge der Felder wird durch die Feldnamen in der Kopfzeile bestimmt.

Alle alphanumerischen Werte in der CSV-Datei müssen immer in doppelte Anführungszeichen eingeschlossen werden, da alphanumerische Werte einige Zeichen wie Trennzeichen (";"), Zeilenumbrüche oder doppelte Anführungszeichen enthalten können. Wenn ein Wert ein eingebettetes (doppeltes) Anführungszeichen enthält, dann muss dieses doppelte Anführungszeichen durch zwei (doppelte) Anführungszeichen repräsentiert werden.

Die Inhalte der Felder werden in denjenigen Typ konvertiert, der für diese Felder in der Schema-DDL definiert wurde und werden in der Datenbank im korrekten Format gespeichert.

In dem Fall, dass aufgrund eines falschen Wertes in der CSV-Datei ein Feldinhalt nicht in den Feldtyp konvertiert werden kann, wird der Fehler 908 ausgegeben. Der aktuelle Satz wird nicht gespeichert und BINILOAD setzt die Verarbeitung mit der nächsten Zeile der CSV-Datei fort.

Fehlt ein Feld des DB-Satzes in der CSV-Datei, so füllt BINILOAD das Feld mit dem Defaultwert auf (X’00’ oder X’ 40’, abhängig vom Datentyp).

Ist ein Feld in der CSV-Datei vorhanden, das im DB-Satztyp nicht existiert und das auch nicht einem Feld des Systeminformationsteils zugeordnet werden kann (wie ‘DB Key Ref’, ‘DB Key RSQ, ‘DB Key Ref set-name’, ‘DB Key RSQ set-name’, ‘Member set-name’), dann weigert sich BINILOAD, die CSV-Datei zu laden.

Die Eingabedatei kann durch BOUTLOAD oder ein beliebiges anderes Tool erstellt werden. Die CSV-Eingabedatei kann eine SAM-Datei oder eine ISAM-Datei (EDT-Format) sein, die Sätze verschiedener Länge im variablen Satzformat (RECFORM=V) enthält.

Der Name der Eingabedatei wird in der INPUT-FILE-Anweisung angegeben.

Um Daten aus einer Eingabedatei im CSV-Format zu laden, muss der DBCOM verfügbar sein.

Am Ende dieses Kapitels finden Sie ein Beispiel für eine BINILOAD-Kommandofolge zum Laden von Daten aus einer Eingabedatei im CSV-Format.

5.1.4 Systemumgebung von BINILOAD

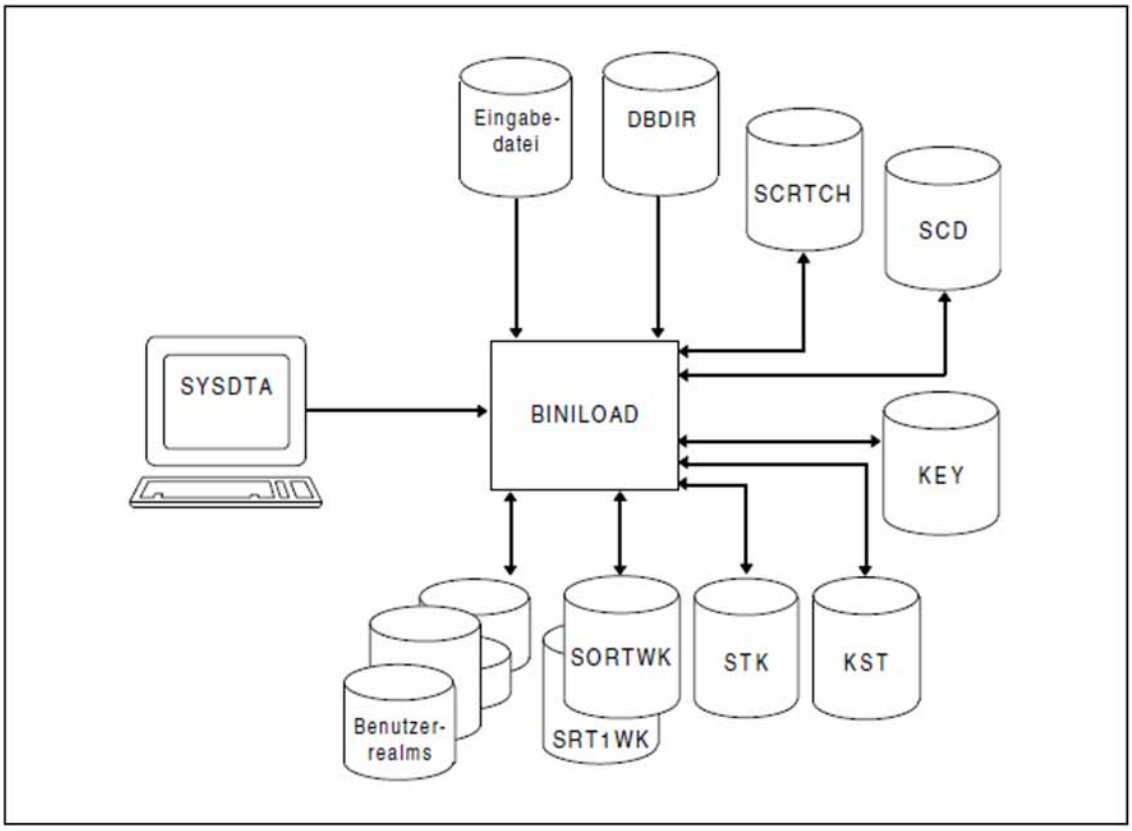

Bild 19: Systemumgebung von BINILOAD

BINILOAD benötigt mehrere Arbeitsdateien, die BINILOAD automatisch auf gemeinschaftlicher Platte in der richtigen Größe mit dem Namen UTI.SAMWORK.*tsn.zeitstempel* anlegt und nach normaler Beendigung des Ladevorgangs wieder löscht.

Die Dateien haben folgende Standard-Linknamen:
SCRTCH1, SCRTCH2, SCRTCH3, SORTWK, SRT1WK, SCD*nnnnn*, STK*nnnnn*, KEY*mmmmm* und KST*nnnnn*.

SCRTCH1	enthält während der Laufzeit eine Folgeversion der Eingabedatei.
SCRTCH2 SCRTCH3	werden verwendet, um den abzuspeichernden Sätzen Platz zuzuweisen.
SORTWK SRT1WK	benötigt der von BINILOAD benutzte SORT für die Sortierung interner Auswertungssätze (siehe Handbuch „SORT (BS2000)“).
SCD*nnnnn*	enthalten während der Laufzeit die SCD-Informationen der Sätze zum Set mit der fünfstelligen Setnummer *setref*, bei denen die zu ladende Satzart Member ist.
STK*nnnnn*	enthalten während der Laufzeit die SEARCH-Key-Informationen der Sätze zum Set mit der fünfstelligen Setnummer *setref*, bei denen die zu ladende Satzart Member ist.
KEY*mmmmm*	enthalten während der Laufzeit die Schlüssel der Sätze zum Key mit der fünfstelligen Schlüsselnummer *keyref*, aus denen die Zugriffstabellen aufzubauen sind.
KST*nnnnn*	enthalten während der Laufzeit Speicherinformationen der Sätze zum Set mit der fünfstelligen Setnummer *setref*, bei denen es keinen vom Anwender definierten Sort-Key gibt. Im BPSIA-Protokoll sind keine Schlüsselnummern *keyref* für den Set ausgewiesen.

Datenbanksicherung

BINILOAD schreibt After-Images, wenn zuvor mit dem Dienstprogramm BMEND für die aktuelle Datenbank AFIM-Logging vereinbart wurde (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BMEND).

ALOG-Dateien

Wenn AFIM-Logging eingeschaltet ist, muss die aktuelle ALOG-Datei vorhanden sein.
Bei einem Fehler auf der ALOG-Datei während des Ablaufs von BINILOAD oder bei einem ALOG-Datei-Überlauf wird das AFIM-Logging abgeschaltet; BINILOAD wird dann ohne ALOG-Datei fortgesetzt und zu Ende geführt. Es entsteht dabei eine Logging-Lücke.

Am Ende des BINILOAD-Laufs wird die ALOG-Datei gewechselt, d.h. es wird eine neue ALOG-Datei eingerichtet.

5.1.5 Anweisungen für BINILOAD

Zum Ablauf von BINILOAD müssen Sie eine Reihe von Anweisungen angeben. BINILOAD kennt vier Arten von Anweisungen:

- Steueranweisungen
- Programmanweisungen
- STORE-Anweisungen
- INSERT-Anweisungen

Anweisungen, die Sie nur wahlweise anzugeben brauchen, sind gekennzeichnet. Es wird empfohlen, alle Anweisungen in der angegebenen Reihenfolge anzugeben, auch wenn Sie Anweisungen nicht benutzen. Für STORE-Anweisungen und INSERT-Anweisungen ist die Reihenfolge zwingend.

Steueranweisungen

steuern den Ablauf des UDS/SQL-Dienstprogramms BINILOAD

Anweisung	Standardwert	Bedeutung
[EXECUTION {WITH | WITHOUT} CHECK.]	WITH	Eingaben prüfen/nicht prüfen
[SORTCORE IS *nnn*.]	150	Größe des Hauptspeichers für das Sortier-/Mischprogramm festlegen

Tabelle 21: Steueranweisungen des BINILOAD

Programmanweisungen

bestimmen Schema, Subschema, Eingabedatei und den Füllgrad von Tabellen

Anweisung	Standardwert	Bedeutung
SCHEMA NAME IS *schemaname*. SUBSCHEMA NAME IS *subschema name*.	-	Name des Schemas und des Subschemas
FILLING IS *nnn* PERCENT.	-	Füllgrad für Tabellenseiten angeben
USER FILE RECORD LENGTH IS *n*.	-	Länge der Eingabesätze in byte angeben
USER FILE BUFFER LENGTH IS *n*.	-	Blocklänge der Eingabedatei; muss ein Vielfaches von 2048 sein.
INPUT FILE NAME '*dateiname*' [FORMAT IS CSV].	-	Dateiname der Eingabedatei
INPUT RECORDNUMBER IS n.	-	keine, wird aus Kompatibilitätsgründen geduldet

Tabelle 22: Programmanweisungen des BINILOAD

In dem Fall, dass FORMAT IS CSV angegeben wurde, ist die Anweisung **USER FILE RECORD LENGTH** nicht erlaubt, da die CSV-Datei eine Datei variabler Länge mit der maximalen Länge von 32752 Bytes ist und die tatsächliche Satzlänge vom Datei-Satz genommen wird. Die Anweisung **USER BUFFER LENGTH** ist ebenfalls nicht erlaubt, es wird die Blocklänge aus den Datei-Eigenschaften verwendet.

STORE-Anweisungen

informieren BINILOAD über die Satzart und deren Zusammenhang mit den Eingabesätzen

Anweisung	Standardwert	Bedeutung
STORE RECORD NAME IS *satzname*.	-	Satzart, die gespeichert werden soll
RECORD-DBKEY IS DISPL IS n, LENGTH IS {4 | 8}	-	Database-Key-Wert vergeben; • Position und Länge des Database-Key-Wertes
RECORD-RSQ IS DISPL IS n, LENGTH IS {3 | 6}	-	Database-Key-Wert vergeben; • Position und Länge der Satzfolgenummer (RSQ) Die zugehörige Satzartnummer (REC-REF) wird von BINILOAD ermittelt.
RECORD-DISPL IS *n*, {DISPL IS *n*,LENGTH IS *n* | VALUE IS '*literal*' }.	-	Datenbanksatz aufbauen zur angegebenen Satzart; • Position und Länge von Feldern dieses Satzes • Zeichenfolge, die in die Datenbanksätze eingefügt werden soll
RECORD-AREA NAME IS *realm name*.	-	Realm, in den die Sätze geladen werden sollen

Tabelle 23: STORE-Anweisungen des BINILOAD

In dem Fall, dass FORMAT IS CSV angegeben wurde, sind die Anweisungen für die Beschreibung des Satzes, RECORD-DBKEY, RECORD-RSQ und RECORD-DISPL, nicht erlaubt. Im Fall von FORMAT IS CSV werden keine festen Verlagerungen verwendet, sondern die Positionen des DB-KEY (Ref and RSQ) werden durch die Kopfzeile ermittelt.

INSERT-Anweisungen

nennen BINILOAD die Sets, in die die Sätze einzufügen sind

Anweisung	Standardwert	Bedeutung
INSERT INTO SET NAME IS *setname*.	-	den Set angeben, in den die Sätze als Membersätze eingefügt werden sollen;
SET ORDER {USING { DISPL IS *n*, LENGTH IS *n* | FIELD NAME IS *field-name* | '*field-name*'} | VIA USER FILE SEQUENCE}.	VIA USER FILE SEQUENCE	Sortierfolge der Sätze innerhalb der Sets mit ORDER IS FIRST, LAST, NEXT, PRIOR, IMMATERIAL; LENGTH spezifiert die Länge des Sortierfeldes FIELD NAME IS ist nur erlaubt wenn FORMAT IS CSV angegeben ist. *field-name* ist ein Feldname in der CSV Headerzeile. Er muss in einfachen Hochkommata angegeben werden, wenn der Name Leerzeichen enthält. Für die Database-Key Felder können die Feldnamen 'DB Key Ref' bzw. 'DB Key RSQ' verwendet werden. Fuer Felder vom Datentyp DBKEY kann der Name 'item-name:DB Key Ref' bzw. 'item-name:DB Key RSQ' verwendet werden.
OWNER CALCKEY IS {DISPL IS *n*, LENGTH IS *n* | VALUE IS '*literal*'}, AREA NAME IS *realmname*.	-	Set-Occurrence auswählen durch Auswählen des Owner: • Position und Länge der CALC-Key-Werte in den Sätzen der Eingabedatei, mit denen der Owner ausgewählt werden soll • Zeichenfolge mit CALC-Key • Name des Realm, in dem der Ownersatz gespeichert ist
OWNER SEARCHKEY IS {DISPL IS *n*, LENGTH IS *n* | VALUE IS '*literal* '}, [VIA SET NAME IS *setname*,] SEARCHKEY TABLE {COLUMN-NR IS *n* | ORDER-NR IS *keyref*}.	-	Set-Occurrence auswählen durch Auswählen des Owner über SEARCH-Key: • Position und Länge der SEARCH-Key-Werte in den Sätzen der Eingabedatei, mit denen der Owner ausgewählt werden soll • Zeichenfolge mit SEARCH-Key-Tabelle • Name des SYSTEM-Set, in dem der Owner Member ist • DBTT-Column-Nr der SEARCH-Key-Tabelle • Schlüsselnummer
OWNER DBKEY IS {DISPL IS *n*, LENGTH IS {4 | 8} | VALUE IS *dbkey* }.	-	Set-Occurrence auswählen durch Auswählen des Owner über seinen Database-Key-Wert: • Position und Länge des Database-Key-Werts in den Sätzen der Eingabedatei, mit denen der Owner ausgewählt werden soll • Zeichenfolge mit Database-Key-Wert

OWNER RSQ IS {DISPL IS *n*, LENGTH IS {3 | 6} | VALUE IS *rsq* }.	-	Set-Occurrence auswählen durch Auswählen des Owner über seinen Database-Key-Wert: • Position und Länge der Satzfolgenummer (RSQ) in den Sätzen der Eingabedatei, mit denen der Owner ausgewählt werden soll • Zeichenfolge mit Satzfolgenummer (RSQ). Die zugehörige Satzartnummer (REC-REF) wird von BINILOAD ermittelt.
OWNER KEY IS DISPL IS *n*, LENGTH IS 1.	-	Position des Feldes in den Eingabesätzen, das angibt, ob der Satz in den SYSTEM-Sets eingehängt werden soll.

Tabelle 24: INSERT-Anweisungen des BINILOAD

In dem Fall, dass FORMAT IS CSV angegeben wurde, kann in der SET ORDER-Anweisung die Sortierfolge durch die Reihenfolge in der Benutzer-Datei festgelegt werden, indem das in item-name spezifizierte Feld verwendet wird. Die Angabe von Displacement ist jedoch nicht erlaubt. Die OWNER-Anweisungen (OWNER CALCKEY, OWNER SEARCHKEY, OWNER DBKEY, OWNER RSQ und OWNER KEY) sind nur erlaubt, wenn FORMAT IS CSV nicht angegeben wurde.

5.1.5.1 EXECUTION (Eingabedaten prüfen/nicht prüfen)

Die Anweisung EXECUTION können Sie wahlweise angeben.

```
EXECUTION { WITH | WITHOUT } CHECK.
```

WITH — BINILOAD prüft vor dem Ändern der Datenbank, ob die Eingabedaten und die Struktur der Datenbank miteinander vereinbar sind und ob genügend Platz in der Datenbank vorhanden ist.

Findet BINILOAD Abweichungen in den Eingabedaten von der Struktur der Datenbank, so gibt BINILOAD entsprechende Meldungen aus und bricht den Lauf ab. Nur wenn Eingabedaten und die Struktur der Datenbank miteinander vereinbar sind, speichert BINILOAD Tabellen und Sätze ab.

Ist nicht genügend Platz in einem Realm vorhanden, so wird dies am Ende des Prüflaufs durch die folgende Ablaufmeldung angezeigt:

`MODIFY-REALM-SIZE <realmname>, DIFFERENCE = n .`

Der Programmlauf wird jedoch nicht abgebrochen, wenn in dem Realm SECONDARY_ALLOCATION > 0 eingestellt ist, da der benötigte Freiplatz dann durch automatische Realmerweiterung beschafft wird (siehe auch Handbuch „Datenbankbetrieb“).

Eine automatische DBTT-Erweiterung durch das Dienstprogramm BINILOAD findet nicht statt.

WITHOUT — BINILOAD unterdrückt den Prüflauf, die Daten der Eingabedatei werden sofort gespeichert.Treten Fehler auf, wird der BINILOAD-Lauf abgebrochen und entsprechende Meldungen ausgegeben; die Datenbank ist dann inkonsistent.

Standardwert:

WITH

> BINILOAD kann nicht den Inhalt der Eingabedatei prüfen.

5.1.5.2 SORTCORE (Größe des Sortierpuffers festlegen)

Die Anweisung SORTCORE können Sie wahlweise angeben.

```
SORTCORE IS nnn.
```

nnn Sie legen die Größe des Speicherplatzes in Einheiten von 4-Kbyte für den Sortierpuffer fest, der dem Dienstprogramm SORT des BS2000 zur Verfügung gestellt wird (siehe Handbuch „SORT (BS2000)“, ALLOC-Anweisung). Das Mengengerüst der zu sortierenden Daten ist das gleiche, das der Größe der Arbeitsdateien mit den Linknamen SORTWK bzw. SRT1WK zugrunde liegt (siehe "Arbeitsdateien einrichten").

Standardwert:150

5.1.5.3 SCHEMA (Name des Schemas angeben)

Die Anweisung SCHEMA können Sie wahlweise angeben.

```
SCHEMA NAME IS schemaname.
```

schemaname

in der Schema-DDL angegebener Name des Schemas

5.1.5.4 SUBSCHEMA (Name des Subschemas angeben)

Die Anweisung SUBSCHEMA müssen Sie angeben.

```
SUBSCHEMA NAME IS subschemaname.
```

subschemaname

in der Subschema-DDL angegebener Name des Subschemas

Mit Hilfe der angegebenen Namen erhält BINILOAD Informationen über die Datenbank, in die die neuen Sätze gespeichert werden sollen.

5.1.5.5 FILLING (Füllgrad von Tabellenseiten festlegen)

Die Anweisung FILLING können Sie wahlweise angeben.

```
FILLING IS nnn PERCENT.
```

nnn legt fest, zu wie viel Prozent Tabellenseiten auf der Stufe 0 gefüllt werden sollen. Zukünftige Einfügungen in diese Datenelemente sind dadurch möglich. Seiten, die nicht für Tabellen verwendet werden, werden vollständig gefüllt.

nnn = 1 ... 100

Für Tabellenseiten auf der Stufe 1 beträgt der Standard-Füllgrad 95%, auf allen darüberliegenden Stufen bleibt jeweils ein Tabelleneintrag frei.

Wenn Sie FILLING weglassen, bleibt auch auf Stufe 0 ein Eintrag frei.

Geben Sie *nnn* zu niedrig an, stellt BINILOAD sicher, dass mindestens ein Eintrag möglich ist.

5.1.5.6 USER RECORD LENGTH (Länge der Eingabesätze angeben)

Die Anweisung USER RECORD LENGTH müssen Sie angeben, wenn die Eingabedatei nicht im CSV-Format ist.

In dem Fall, dass die Eingabedatei im CSV-Format ist, ist die Anweisung USER FILE RECORD LENGTH nicht erlaubt, da die CSV-Datei eine Datei variabler Länge mit einer Maximallänge von 32752 Bytes ist und die tatsächliche Satzlänge jedesmal aus dem Satz in der Datei genommen wird.

```
USER FILE RECORD LENGTH IS n.
```

n Gesamtlänge eines Satzes der Eingabedatei in byte.
Die Sätze der Eingabedatei können neben den Feldinhalten der Sätze, die in die Datenbank gespeichert werden sollen, zusätzlich Benutzerinformation und Steuerinformation enthalten.

Ist für die Eingabedatei das Satzformat „variable“ (RECFORM=V) vereinbart, so enthält die Längenangabe die Satzlänge ohne das Satzlängenfeld (RECSIZE - 4).

n > 0

5.1.5.7 USER BUFFER LENGTH (Blocklänge der Eingabedatei angeben)

Die Anweisung USER BUFFER LENGTH müssen Sie nicht angeben, wenn die Eingabedatei mit fester Satzlänge (RECFORM=F) erstellt wurde.

In dem Fall, dass die Eingabedatei im CSV-Format ist, ist die Anweisung USER BUFFER LENGTH nicht erlaubt. Es wird die in der Datei festgelegte Blocklänge verwendet.

```
USER FILE BUFFER LENGTH IS n.
```

n Blocklänge der Eingabedatei in byte.
BINILOAD legt einen Puffer in der angegebenen Länge an.

n muss ein Vielfaches von 2048 sein.

5.1.5.8 INPUT FILE (Dateiname der Eingabedatei angeben)

Die Anweisung INPUT FILE müssen Sie angeben.

```
INPUT FILE NAME IS 'dateiname' [FORMAT IS CSV].
```

'*dateiname*'

Name der Eingabedatei mit den zu speichernden Sätzen. Die Datei kann eine SAM- oder ISAM-Datei (EDT-Format) sein; neben Sätzen variabler Länge (RECFORM=V) sind auch Sätze fester Länge (RECFORM=F) erlaubt.

Für den Fall, dass FORMAT IS CSV angegeben wurde, muss es sich um eine SAM-Datei mit Sätzen variabler Länge handeln (RECFORM=V).

dateiname müssen Sie im Literalformat angeben, da der Name mehrstufig qualifiziert sein kann.

5.1.5.9 STORE RECORD (Satzart angeben)

Die Anweisung STORE RECORD müssen Sie angeben.

```
STORE RECORD NAME IS satzname.
```

satzname

Name der Satzart, deren Sätze in die Datenbank gespeichert werden sollen. Der Name muss in dem zugehörigen Schema und Subschema definiert sein.

5.1.5.10 RECORD-DBKEY (Database-Key-Wert des Satzes vergeben)

Wenn Sie für jeden Satz der bei STORE RECORD (siehe "STORE RECORD (Satzart angeben)") angegebenen Satzart den Database-Key-Wert explizit vergeben wollen, müssen Sie entweder die Anweisung RECORD-DBKEY oder die Anweisung RECORD-RSQ angeben:

- Bei RECORD-DBKEY geben Sie den vollständigen Database-Key-Wert in der Eingabedatei an.
- Bei RECORD-RSQ geben Sie nur die Satzfolgenummer (RSQ) in der Eingabedatei an. BINILOAD ermittelt dann den Database-Key-Wert des Eingabesatzes mit Hilfe dieser RSQ und der Satzartnummer (REC-REF) der Satzart, die Sie bei STORE RECORD angegeben haben.

Sollen die Database-Key-Werte entsprechend der Reihenfolge der Sätze in der Eingabedatei vergeben werden, so ist die Angabe der Anweisungen RECORD-DBKEY bzw. RECORD-RSQ optional.

In dem Fall, dass die Eingabedatei im CSV-Format ist, sind die Anweisungen RECORD-DBKEY oder RECORD-RSQ nicht erlaubt.

Anweisung RECORD-DBKEY

```
RECORD-DBKEY IS DISPL IS n, LENGTH IS {4 | 8}
```

DISPL IS *n*

gibt die Position des zu vergebenden Database-Key-Wertes im Eingabesatz relativ zum Anfang des Eingabesatzes an.

LENGTH IS {4 | 8}

Die Länge eines Database-Key-Wertes beträgt immer 4 byte oder 8 byte.
8 byte lange Database-Key-Werte mit einer Satzartnummer (REC-REF) > 254 und/oder einer Satzfolgenummer (RSQ) > 2^{24}-1 können Sie nur bei der Eingabe in Datenbanken mit 4000 byte oder 8096 byte Seitenlänge verwenden.

Anweisung RECORD-RSQ

```
RECORD-RSQ IS DISPL IS n, LENGTH IS {3 | 6}
```

DISPL IS *n*

gibt die Position der zu vergebenden Satzfolgenummer (RSQ) im Eingabesatz relativ zum Anfang des Eingabesatzes an.

LENGTH IS {3 | 6}

Die Länge einer Satzfolgenummer (RSQ) beträgt immer 3 byte oder 6 byte.
6 byte lange Satzfolgenummern > 2^{24}-1 können Sie nur bei der Eingabe in Datenbanken mit 4000 byte oder 8096 byte Seitenlänge verwenden.

5.1.5.11 RECORD-DISPL (Datenbanksatz aufbauen)

Die Anweisung RECORD-DISPL müssen Sie angeben, wenn die Sätze der Eingabedatei neben den Datenbanksätzen noch Benutzerinformationen und Steuerinformationen oder Felder für andere Satzarten enthalten, bzw. wenn Felder verschoben werden sollen.

Sie müssen sie mehrfach angeben, wenn mehrere Felder aus dem Eingabesatz gezielt an die zugehörigen Stellen im Datenbanksatz übertragen werden sollen.

Die Anweisung können Sie wahlweise angeben, wenn die Sätze der Eingabedatei identisch sind mit den Sätzen, die in die Datenbank gespeichert werden sollen; bei Satzformat „variable“ entspricht der Datenbanksatz dem Datenteil ohne Satzlängenfeld.

In dem Fall, dass die Eingabedatei im CSV-Format ist, ist die Anweisung RECORD-DISPL nicht erlaubt.

```
RECORD-DISPL IS n, {DISPL IS n,LENGTH IS n | VALUE IS 'literal'}.
```

RECORD-DISPL IS *n*

gibt die Position (relativ zum Satzanfang) des zu übertragenden Feldes im Datenbanksatz an.

DISPL IS *n*

gibt die Position (relativ zum Satzanfang) des zu übertragenden Feldes im Eingabesatz an.
Bei variabel langen Sätzen darf das Satzlängenfeld nicht berücksichtigt werden.

LENGTH IS *n*

gibt die Länge des zu übertragenden Feldes an.

VALUE IS ’*literal*’

gibt einen Wert an, der in jeden gespeicherten Satz an der Stelle eingefügt wird, die durch das *n* der RECORD-DISPL-Anweisung angegeben wurde.

Das Literal kann sein:

- eine Zeichenfolge, z.B. ’datum’ (max. 64 byte)
- eine sedezimale Zeichenfolge, z.B. ’014F’X, ’FFFF’X usw. (max. 32 byte)

Wenn ein Apostroph in einer Zeichenfolge enthalten sein soll, müssen Sie zwei Apostrophe angeben.

Die RECORD-DISPL-Anweisung mit der VALUE-Klausel können Sie maximal fünfmal angeben.

Beispiel zur RECORD-DISPL-Anweisung

RECORD-DISPL IS 0

0 bezeichnet die Anzahl der Bytes zwischen dem Anfang des Datenbanksatzes und dem ersten Byte, an das das Feld übertragen werden soll

DISPL IS 3

3 bezeichnet die Anzahl der Bytes (Abstand) zwischen dem Satzanfang des Eingabesatzes und dem ersten Byte des zu übertragenden Feldes

LENGTH IS 6

6 bezeichnet die Länge des zu übertragenden Feldes

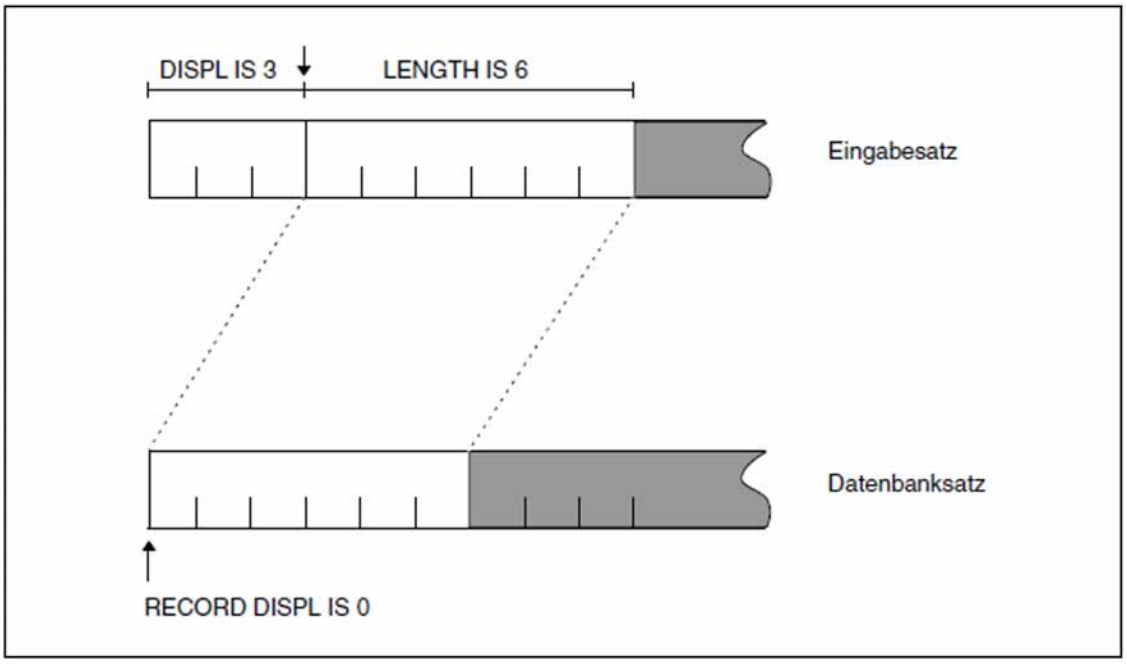

Bild 20: Eingabesatz und Datenbanksatz mit dem zu übertragenden Feld

Jede RECORD-DISPL-Anweisung erzeugt oder erweitert eine MOVE-Anweisung. Es werden Teile des Eingabesatzes eingefügt bzw. bei VALUE Zeichenfolgen eingefügt.
Werden mehrere RECORD-DISPL-Anweisungen angegeben, gilt die Reihenfolge der Anweisungen. Dabei können Teile des Satzes, die durch vorangegangene RECORD-DISPL-Anweisungen eingefügt wurden, überschrieben werden.
Die RECORD-DISPL-Anweisung darf sich nicht auf eine Position außerhalb des Datenbanksatzes beziehen. Es wird nicht geprüft, ob der Wert des Literals oder der Typ des Eingabesatzes dem Element im Datenbanksatz entspricht, wie er im Schema definiert ist. Eine Konvertierung wird von BINILOAD ebenfalls nicht durchgeführt.

5.1.5.12 RECORD-AREA (Realm angeben)

Die Anweisung RECORD-AREA müssen Sie angeben, wenn die WITHIN-Klausel der Schema-DDL mehr als einen Realm-Namen enthält und die zu speichernde Satzart nicht Membersatzart einer verteilbaren Liste ist.

In folgenden Fällen können Sie die Anweisung RECORD-AREA wahlweise angeben:

- Wenn die WITHIN-Klausel der Schema-DDL nur einen Realm-Namen enthält
- Wenn die zu speichernde Satzart Member einer verteilbaren Liste ist:

  Verzichten Sie bei verteilbaren Listen auf die Angabe eines Realms, dann speichert BINILOAD die Sätze in etwa gleichmäßig in allen Realms, die in der WITHIN-Klausel der Schema-DDL angegeben sind.
  Geben Sie bei verteilbaren Listen mit der Anweisung RECORD-AREA einen Realm an, dann muss es sich um den Tabellenrealm handeln. Die Sätze werden dann in diesen Realm gespeichert. Die zugehörige Liste bleibt verteilbar.

Wenn Sie mit der Anweisung RECORD-AREA einen Realm angeben, so muss dieser Realm im Subschema definiert sein. Ist die zu speichernde Satzart Membersatzart einer verteilbaren Liste, so müssen alle Realms der WITHIN-Klausel der Schema-DDL im Subschema definiert sein.

```
RECORD-AREA NAME IS realmname.
```

realmname

Name des Realms, in den die Sätze geladen werden sollen.

> **¡** Berücksichtigen Sie die Angaben der SSL!
> Ist die zu speichernde Satzart Member eines Set mit MODE IS LIST ohne den Zusatz DETACHED WITHIN *realmname* oder eines Set mit PLACEMENT OPTIMIZATION, so müssen sich die Owner für die zu speichernden Set-Occurrences ebenfalls in dem spezifizierten Realm für die Membersatzart befinden.

5.1.5.13 INSERT (Set angeben)

Die Angabe der Anweisung INSERT ist abhängig von der Mitgliedschaft der Membersätze im Set (siehe Handbuch „Entwerfen und Definieren").

- Standard-Set MANDATORY AUTOMATIC
  Sie müssen die Anweisung angeben, anschließend muss die OWNER-Anweisung folgen.
- Standard-Set OPTIONAL oder MANUAL
  - Sie müssen die Anweisung angeben, wenn alle Sätze oder ein Teil von ihnen eingehängt werden sollen, anschließend muss die OWNER-Anweisung folgen.
  - Sie dürfen die Anweisung nicht angeben, wenn keiner der Sätze in die Set-Occurrence eingehängt werden soll; die OWNER-Anweisung entfällt.
- SYSTEM-Set MANDATORY AUTOMATIC
  Sie müssen die Anweisung angeben, die anschließende OWNER-Anweisung entfällt.
- SYSTEM-Set OPTIONAL oder MANUAL
  - Sie müssen die Anweisung angeben, wenn nur ein Teil der Sätze eingehängt werden soll; anschließend muss die OWNER-Anweisung folgen.
  - Sie müssen die Anweisung angeben ohne nachfolgende OWNER-Anweisung, wenn alle Membersätze eingehängt werden sollen.
  - Sie dürfen die Anweisung nicht angeben, wenn keiner der Sätze in die SET-Occurrence des SYSTEM-Set eingehängt werden soll; die OWNER-Anweisung entfällt.
  - Der Set muss im angegebenen Subschema definiert sein.
- SYSTEM-Set IMPLICIT
  Sie dürfen die Anweisung nicht angeben.

```
INSERT INTO SET NAME IS setname.
```

setname

gibt an, in welchen Set die Sätze der Eingabedatei als Member eingehängt werden sollen

> Die INSERT- und OWNER-Anweisungen sind anzugeben, wenn BOUTLOAD beim Entladen die Set Connection Data (SCD) mit erzeugt und anschließend BINILOAD die alten Set-Mitgliedschaften wiederherstellen soll.

5.1.5.14 SET ORDER (Sortierfolge angeben)

Die Anweisung können Sie wahlweise angeben, wenn in der ORDER-Klausel der Schema-DDL die Sortierung innerhalb der Sets definiert wurde mit FIRST, LAST, NEXT, PRIOR oder IMMATERIAL (siehe Handbuch „Entwerfen und Definieren") und die Reihenfolge der Sätze in der Set-Occurrence nicht derjenigen in der Eingabedatei entspricht.

In diesem Fall können Sie die Reihenfolge innerhalb der Set-Occurrence beim Laden mit BINILOAD festlegen, indem Sie in den Eingabesätzen je ein Sortierfeld festlegen. Der Inhalt dieses Feldes wird benutzt, um die Membersätze in aufsteigender Reihenfolge zu sortieren.

Die Anweisung SET ORDER müssen Sie nicht angeben, wenn die zu speichernden Datenbanksätze in der Reihenfolge in der Eingabedatei vorkommen, wie sie in die Set-Occurrence eingehängt werden sollen.

Sie dürfen sie nicht angeben, wenn in der ORDER-Klausel der Schema-DDL die Sortierung innerhalb des Set definiert wurde mit SORTED, SORTED INDEXED.

```
SET ORDER {USING {DISPL IS n, LENGTH IS n | FIELD NAME IS item-name} | VIA USER
FILE SEQUENCE}.
```

USING DISPL IS *n*

gibt die Position (relativ zum Satzanfang) des Sortierfeldes im Eingabesatz an
In dem Fall, dass die Eingabedatei im CSV-Format ist, ist die Option **USING DISPL IS n** nicht erlaubt.

USING FIELD NAME IS *item-name*

gibt den Namen des Sortierfeldes im Eingabesatz an. Ist nur erlaubt, wenn die Eingabedatei im CSV-Format ist.

VIA USER FILE SEQUENCE

die Reihenfolge der Sätze in der Eingabedatei wird übernommen als Reihenfolge für die Set-Occurrences

Standardwert:

VIA USER FILE SEQUENCE

Der SET ORDER-Anweisung muss die zugehörige INSERT-Anweisung vorausgehen.

5.1.5.15 OWNER (Owner bestimmen)

Wenn die Eingabedatei nicht im CSV-Format ist, dann müssen Sie die Anweisung OWNER bei allen Sets (Ausnahmen bei SYSTEM-Sets) angeben unabhängig davon, ob die Sets MANUAL- oder AUTOMATIC-Member haben, wenn zuvor eine INSERT-Anweisung gegeben wurde.
Ist die Eingabedatei im CSV-Format, dann wird der Owner durch seinen DB-KEY anhand der Kopfzeile ermittelt.

BINILOAD kann den Owner bestimmen, wenn Sie in einem der Formate 1 bis 3 folgende Werte angeben (bei SYSTEM-Sets gilt Format 4):

Wert	Bedingung	Format
CALC-Key	-	Format 1
ASC-/DESC-Key SEARCH-Key	wenn der Owner gleichzeitig Member in einem SYSTEM-Set ist	Format 2
	wenn der SEARCH-Key auf Satzartebene definiert wurde	
DB-Key	-	Format 3

Alle Schlüsselwerte können Sie in den Eingabesätzen, als Inhalt eines Feldes, oder in der OWNER-Anweisung für diesen Set, als Literal, angeben. Wenn der Schlüsselwert als Literal angegeben wird, so werden alle Sätze der Eingabedatei dem gleichen Owner zugeordnet.

Wenn die Ownersatzart Membersatzart einer verteilbaren Liste ist, müssen die Realms der DDL-WITHIN-Klausel der Ownersatzart im Subschema definiert sein.

> ¡ Wenn DUPLICATES ARE ALLOWED in der Schema-DDL angegeben wurde und CALC-, ASC-/DESC-, oder SEARCH-Keys verwendet wurden, können gleiche Schlüsselwerte auftreten. In diesem Fall kann nicht vorausgesagt werden, welche Ownersätze BINILOAD auswählt.
>
> Sollen bei einem MANUAL-Set oder einem OPTIONAL-Set bestimmte Membersätze nicht eingehängt werden, so ist das Feld für die Ownerauswahl in der Eingabedatei mit HIGH-VALUE zu belegen.

Format 1: Owner durch CALC-Key bestimmen

In dem Fall, dass die Eingabedatei im CSV-Format ist, ist die Anweisung OWNER CALCKEY nicht erlaubt.

```
OWNER CALCKEY IS {DISPL IS n, LENGTH IS n | VALUE IS 'literal'},
AREA NAME IS realmname .
```

DISPL IS *n*

gibt die Position des Feldes im Eingabesatz an, welches den CALC-Key enthält

LENGTH IS *n*

gibt die Länge des Feldes an, welches den CALC-Key enthält (Länge des CALC-Keys)

VALUE IS '*literal*'

gibt den CALC-Key an, der den Owner für alle Sätze der Eingabedatei auswählt

realmname

bezeichnet einen Realm, der in der DDL-WITHIN-Klausel der Ownersatzart angegeben wurde.
Wenn die Ownersatzart Membersatzart einer verteilbaren Liste ist, müssen Sie hier deren Tabellenrealm angegeben, da sich dort der indirekte CALC-Bereich befindet.

Den AREA-Eintrag im Format 1 müssen Sie immer angeben.

Format 2: Owner durch SEARCH-Key bestimmen

In dem Fall, dass die Eingabedatei im CSV-Format ist, ist die Anweisung OWNER SEARCHKEY nicht erlaubt.

```
OWNER SEARCHKEY IS {DISPL IS n, LENGTH IS n | VALUE IS 'literal'},
[ VIA  SET NAME IS setname ,]
     SEARCHKEY TABLE { COLUMN-NR IS n | ORDER-NR IS keyref }.
```

DISPL IS *n*

Position des Feldes im Eingabesatz, das den SEARCH-Key enthält

LENGTH IS *n*

Länge des Feldes, das den SEARCH-Key enthält

VALUE IS '*literal*'

SEARCH-Key-Wert, der den Owner für alle Sätze der Eingabedatei bestimmt

VIA SET NAME IS

Name des SYSTEM-Set, in dem die Ownersatzart Member ist,

setname

darf nicht für solche SYSTEM-Sets angegeben werden, die auf Grund eines RECORD-SEARCH-Key vom DDL-Compiler angelegt werden

SEARCHKEY TABLE

Die Auswahl des Owner erfolgt über den ASC-/DESC- oder SEARCH-Key. Der Schlüsselwert muss in einer Tabelle enthalten sein. Daher können Sie bei SYSTEM-Sets mit MODE IS CHAIN (siehe Handbuch „Entwerfen und Definieren") den ASC-/DESC-Key nur verwenden, wenn in der Schema-DDL ORDER IS SORTED INDEXED angegeben wurde.

COLUMN-NR IS *n*

DBTT-Column-Nr der entsprechenden SEARCH-Key- oder Sort-Key-Tabelle (siehe Handbuch „Sichern, Informieren und Reorganisieren", SIA PRINT REPORT).

ORDER-NR IS *keyref*

Diesen Zusatz müssen Sie angeben, wenn der SEARCH-Key des Owner ein CALC-SEARCH-Key ist. Mit *keyref* geben Sie die Schlüsselnummer an; sie ergibt sich aus der Reihenfolge, in der die Schlüssel innerhalb der Satzart- oder Setbeschreibung in der DDL definiert wurden.

Dieser Zusatz kann auch an Stelle des COLUMN-NR-Zusatzes verwendet werden.

Format 3: Owner durch Database Key bestimmen

In dem Fall, dass die Eingabedatei im CSV-Format ist, ist die Anweisung OWNER DBKEY nicht erlaubt.

Mit den Anweisungen OWNER DBKEY und OWNER RSQ können Sie den Ownersatz über seinen Database-Key-Wert bestimmen:

- Bei OWNER DBKEY geben Sie den vollständigen Database-Key-Wert des Ownersatzes an.
- Bei OWNER RSQ geben Sie die Satzfolgenummer (RSQ) des Ownersatzes an. BINILOAD ermittelt dann den Database-Key-Wert des Ownersatzes mit Hilfe dieser RSQ und der Satzartnummer (REC-REF), die der Owneratzart desjenigen Set zugeordnet ist, den Sie in der zuletzt angegebenen INSERT-Anweisung (siehe "INSERT (Set angeben)") genannt haben.

Anweisung OWNER DBKEY

```
OWNER DBKEY IS {DISPL IS n, LENGTH IS {4 | 8} | VALUE IS dbkey}.
```

DISPL IS *n*

Position des Feldes in den Eingabesätzen, das den Database-Key-Wert enthält

LENGTH IS {4 | 8}

Länge des Database-Key-Wertes.
Den Database-Key-Wert müssen Sie in der angegebenen Länge in den Sätzen der Eingabedatei bereitstellen: Das Feld, das den Ownersatz bestimmt, muss den Database-Key-Wert binär dargestellt enthalten. Zur binären Darstellung von Database-Key-Werten siehe Handbuch „Entwerfen und Definieren".
Wenn der Database-Key-Wert des Ownersatzes eine Satzartnummer (REC-REF) > 254 und/oder eine Satzfolgenummer (RSQ) > 2^{24} -1 enthält, müssen Sie „LENGTH IS 8“ angeben.

VALUE IS *dbkey*

Database-Key-Wert, der den Ownersatz für alle Sätze der Eingabedatei auswählt.
Den Database-Key-Wert geben Sie wie folgt an:

Satzartnummer (REC-REF) : Satzfolgenummer (RSQ)

Für den Wertebereich von REC-REF und RSQ gilt:

- im Fall „LENGTH IS 4“: 1 < REC-REF <=254 und 0 < RSQ <=2^{24}-1
- im Fall „LENGTH IS 8“: 1 < REC-REF <= 2^{15}-1 und 0 < RSQ <= 2^{31}-1

Beispiel zur Eingabe des Database-Key-Wertes

Den Database-Key-Wert mit der REC-REF = 22 und der RSQ = 10 596 können Sie wie folgt angeben:

1. In der VALUE IS-Klausel:

```
VALUE IS  22 : 10596
          |       |
       REC-REF   RSQ
```

2. In der Eingabedatei:

- im Fall „LENGTH IS 4“: X’16002964’
- im Fall „LENGTH IS 8“: X’0016000000002964’

Anweisung OWNER RSQ

In dem Fall, dass die Eingabedatei im CSV-Format ist, ist die Anweisung OWNER RSQ nicht erlaubt.

```
OWNER RSQ IS {DISPL IS n, LENGTH IS {3 | 6} | VALUE IS rsq}.
```

DISPL IS *n*

Position des Feldes in den Eingabesätzen, das die Satzfolgenummer (RSQ) enthält

LENGTH IS {3 | 6}

Länge der Satzfolgenummer (RSQ).
Die RSQ müssen Sie in der angegebenen Länge in den Sätzen der Eingabedatei bereitstellen: Das Feld, das den Ownersatz bestimmt, muss die RSQ binär dargestellt enthalten.
Wenn die RSQ des Ownersatzes größer ist als 2^{24} -1, müssen Sie „LENGTH IS 6“ angeben.

VALUE IS *rsq*

Satzfolgenummer (RSQ), die den Ownersatz für alle Sätze der Eingabedatei auswählt. *rsq* geben Sie an als ganze Zahl mit folgendem Wertebereich an:

- im Fall „LENGTH IS 3“: 0 < RSQ <= 2^{24} -1
- im Fall „LENGTH IS 6“: 0 < RSQ <= 2^{31} -1

Beispiel zur Eingabe der Satzfolgenummer (RSQ)

Die RSQ = 10 596 können Sie wie folgt angeben:

1. In der VALUE IS-Klausel:

```
VALUE IS 10596
          |
         RSQ
```

2. In der Eingabedatei:

- im Fall „LENGTH IS 3“: X’002964’
- im Fall „LENGTH IS 6“: X’000000002964’

Format 4: Set-Mitgliedschaft im SYSTEM-Set bestimmen

In dem Fall, dass die Eingabedatei im CSV-Format ist, ist die Anweisung OWNER KEY nicht erlaubt.

```
OWNER KEY IS DISPL IS n, LENGTH IS 1
```

DISPL IS *n*

Position des Feldes in den Eingabesätzen, das angibt, ob der Satz in den SYSTEM-Sets eingehängt werden soll

LENGTH IS 1

die Länge des Feldes ist immer 1

einhängen: X’00’ (LOW-VALUE)
nicht einhängen: X’FF’ (HIGH-VALUE)

Geben Sie die OWNER-Anweisung nicht an, werden alle Sätze in den SYSTEM-Set eingehängt.

5.1.6 Kommandofolge zum Starten von BINILOAD

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 [/CREATE-FILE FILE-NAME=eingabebanddatei,...]
03 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
04 /START-UDS-BINILOAD
05 biniload-anweisungen
06 END
```

03 Die angegebene Version von BINILOAD wird ausgewählt.
Die Angabe der Version wird generell empfohlen, da mehrere UDS/SQL-Versionen parallel installiert sein können.

04 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen BINILOAD gestartet werden.

> **i** Die BINILOAD-Anweisungen werden über SYSDTA eingelesen! Dabei kann es sich auch um eine Datei handeln, die von BOUTLOAD erzeugt wurde.

5.1.7 Arbeitsdateien einrichten

Wollen Sie die Arbeitsdateien explizit einrichten, so müssen Sie die entsprechenden CREATE-FILE-Kommandos angeben. Geben Sie einen zu geringen Speicherplatzwert mit SPACE an, so wird er intern durch BINILOAD korrigiert.

```
/CREATE-FILE FILE-NAME=arbeitsdatei-n [,SUPPORT ...]

/ADD-FILE-LINK LINK-NAME={SCRTCH1 | SCRTCH2 |SCRTCH3 |SCDnnnnn | STKnnnnn |
                          KEYnnnnn | KSTnnnnn | SORTWK | SRT1WK},
                          FILE-NAME=arbeitsdatei-n[,ACCESS-METHOD={PAM | SAM}]
```

arbeitsdatei-n beliebiger Name der Arbeitsdatei

SUPPORT Mit der SPACE-Angabe des SUPPORT-Operanden können Sie die Speicherplatzgröße festlegen.

ACCESS-METHOD

Die Arbeitsdateien SORTWK und SRT1WK werden als PAM-Dateien angelegt, die übrigen Arbeitsdateien als SAM-Dateien.

Linknamen der Arbeitsdateien

SCRTCH1
SCRTCH2
SCRTCH3
SCD*nnnnn*
STK*nnnnn*
KEY*nnnnn*
KST*nnnnn*
SORTWK
SRT1WK

Platzbedarf für die Arbeitsdateien ermitteln

Die Primärzuweisung für die Arbeitsdateien sollte sich am Mengengerüst der zwischenzuspeichernden Daten orientieren. Es sollte immer eine angemessene Sekundärzuweisung erfolgen, für den Fall, dass der Speicherplatz erweitert werden muss.

Mit Hilfe der nachfolgend dargestellten Formeln können Sie den Platzberdarf für die einzelnen Arbeitsdateien näherungsweise ermitteln.

SCRTCH1

(*gesamtschlüssellänge* + 12) x *anzahl der eingabesätze* Bytes

gesamtschlüssellänge:
ist die Gesamtlänge der folgenden Schlüssel:

- Schlüssel, durch welche die Owner des Set ausgewählt werden (CALC-Keys, ASC-/DESC-Keys, SEARCH-Keys oder Database Keys)
- Schlüssel, die nicht zu einem Set gehören (CALC-Keys)
- Schlüssel zu allen Sets, in die diese Sätze eingefügt werden sollen (ASC-/DESC-Keys, SEARCH-Keys).

SCRTCH2

```
12 x anzahl der eingabesätze Bytes
```

SCRTCH3

```
3 x anzahl der eingabesätze Bytes
```

SCD*nnnnn*

bei 2048 byte Seitenlänge:

```
40 x anzahl der eingabesätze Bytes
```

bei 4000/8096 byte Seitenlänge:

```
50 x anzahl der eingabesätze Bytes
```

STK*nnnnn*

bei 2048 byte Seitenlänge:

```
(8 + satzlänge_1) x anzahl der eingabesätze Bytes
```

bei 4000/8096 byte Seitenlänge:

```
(12 + satzlänge_1) x anzahl der eingabesätze Bytes
```

satzlänge_1:
satzlänge_1 ist die Summe aus der Schlüssellänge aller SEARCH-Keys im Set mit der SET-REF *nnnnn*.

KEY*nnnnn* und bei SEARCH-Key

bei 2048 byte Seitenlänge:

```
(16 + schlüssellänge_1) x anzahl der eingabesätze Bytes
```

bei 4000/8096 byte Seitenlänge:

```
(24 + schlüssellänge_1) x anzahl der eingabesätze Bytes
```

schlüssellänge_1:
Schlüssellänge des Schlüssels mit der KEY-REF *nnnnn*

KEY*mmmmm* und KST*nnnnn* bei SORT-Key

bei 2048 byte Seitenlänge:

```
(schlüssellänge_1 + 12 + schlüssellänge_2) x
anzahl der eingabesätze Bytes
```

bei 4000/8096 byte Seitenlänge:

```
(schlüssellänge_1 + 16 + schlüssellänge_2) x
anzahl der eingabesätze Bytes
```

schlüssellänge_1:
Länge des Schlüssels durch den der Owner des Set bestimmt wird.
Dies können sein:
CALC-Keys, ASC-/DESC-Keys, SEARCH-Keys oder Database Keys.

schlüssellänge_2:
Schlüssellänge des Schlüssels mit der KEY-REF*mmmmm*
Im Fall KST*nnnnn* mit der SET-REF *nnnnn* gilt *schlüssellänge_1* = 0.

SORTWK und SRT1WK

Die beiden Arbeitsdateien mit den Linknamen SORTWK und SRT1WK werden vom SORT benötigt, wenn der virtuelle Speicher für die Vorsortierung nicht ausreicht. Die Primärzuweisung sollte sich am Mengengerüst der zu sortierenden Daten orientieren unter Berücksichtigung des von SORT empfohlenen Sicherheitsfaktors (siehe das Thema „Arbeitsdateien“ im Handbuch „SORT (BS2000)“). Es sollte immer eine angemessene Sekundärzuweisung erfolgen, für den Fall, dass der Speicherplatz erweitert werden muss.

Das Mengengerüst der zu sortierenden Daten ergibt sich aus der Formel:

`(`*`satzlänge_2`*`+SCD+12) x `*`anzahl der eingabesätze`* `Bytes`

satzlänge_2:
ist die Länge des Datenbanksatzes

SCD:
Länge der Set Connection Data. Diesen Wert können Sie dem BPSIA-Protokoll entnehmen unter der Überschrift RECORD-INFORMATION in der Spalte SYSINFO (siehe Handbuch „Sichern, Informieren und Reorganisieren“)

Wenn der Ladevorgang beendet ist, bleibt es Ihnen überlassen, die Arbeitsdateien wieder zu löschen, falls diese explizit eingerichtet wurden.

5.1.8 Beispiel zu BINILOAD

Die Satzart ARTIKEL wird in die Datenbank VERSAND gespeichert. Als Eingabedatei wird die von BOUTLOAD erzeugte Datei VERSAND.REC00009.00005 verwendet.

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=VERSAND.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BINILOAD
***** START       BINILOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:38
EXECUTION WITH CHECK.
SCHEMA NAME IS ARTIKELVERSAND
SUBSCHEMA NAME IS ADMIN
USER FILE RECORD LENGTH IS  112
USER FILE BUFFER LENGTH IS 8192
INPUT FILE 'VERSAND.REC00009.00005                              '
INPUT RECORDNUMBER IS      55
STORE RECORD NAME IS ARTIKEL
RECORD-DBKEY IS DISPL IS 0 , LENGTH IS 8
RECORD-DISPL IS 0 , DISPL IS   25 , LENGTH IS   87
RECORD-AREA NAME IS KLEIDUNG
INSERT INTO SET NAME IS BESTELLANGABEN
OWNER DBKEY IS DISPL IS    8 , LENGTH IS 8
INSERT INTO SET NAME IS MIN-BESTAND-ERREICHT
OWNER KEY IS DISPL IS   16 , LENGTH IS 1
INSERT INTO SET NAME IS LIEFERBARE-ARTIKEL
OWNER DBKEY IS DISPL IS   17 , LENGTH IS 8
END
BEGIN CHECK-RUN
*** DATE AND TIME 2017-06-28  11:26:38
BEGIN ALLOCATION
*** DATE AND TIME 2017-06-28  11:26:38

SET_NAME: BESTELLANGABEN
SET_REF:     7
SORTKEY TABLE, DBTT_COLUMN_NR:     1
*** ICRELES: MOVE_ROUTINE SORTKF CREATED
CALCKEY TABLE - INDIRECT
*** DATE AND TIME 2017-06-28  11:26:38
BEGIN TABLE-PROCESSOR
*** DATE AND TIME 2017-06-28  11:26:38

SET_NAME: MIN-BESTAND-ERREICHT
SET_REF:     8
SORTKEY TABLE, DBTT_COLUMN_NR:     1
BEGIN TABLE-PROCESSOR
*** DATE AND TIME 2017-06-28  11:26:38

SET_NAME: LIEFERBARE-ARTIKEL
SET_REF:    12
SORTKEY TABLE, DBTT_COLUMN_NR:     1
BEGIN TABLE-PROCESSOR
*** DATE AND TIME 2017-06-28  11:26:38
SEARCHKEY TABLE, DBTT_COLUMN_NR:     2
INDIRECT CALC-SEARCH-KEY BUCKETS
INDIRECT CALC-SEARCH-KEY BUCKETS
*** NO ERRORS DETECTED DURING CHECK-RUN
END CHECK-RUN
*** DATE AND TIME 2017-06-28  11:26:38
```

```
BEGIN ALLOCATION
*** DATE AND TIME 2017-06-28  11:26:38
*** DATABASE IS IN USE

SET_NAME: BESTELLANGABEN
SET_REF:     7
SORTKEY TABLE, DBTT_COLUMN_NR:     1
*** ICRELES: MOVE_ROUTINE SORTKF CREATED
CALCKEY TABLE - INDIRECT
*** DATE AND TIME 2017-06-28  11:26:39
BEGIN TABLE-PROCESSOR
*** DATE AND TIME 2017-06-28  11:26:39

SET_NAME: MIN-BESTAND-ERREICHT
SET_REF:     8
SORTKEY TABLE, DBTT_COLUMN_NR:     1
BEGIN TABLE-PROCESSOR
*** DATE AND TIME 2017-06-28  11:26:39

SET_NAME: LIEFERBARE-ARTIKEL
SET_REF:    12
SORTKEY TABLE, DBTT_COLUMN_NR:     1
BEGIN TABLE-PROCESSOR
*** DATE AND TIME 2017-06-28  11:26:39
SEARCHKEY TABLE, DBTT_COLUMN_NR:     2
INDIRECT CALC-SEARCH-KEY BUCKETS
INDIRECT CALC-SEARCH-KEY BUCKETS
BEGIN STORE DB-RECORD
*** DATE AND TIME 2017-06-28  11:26:39
STORING DATABASE RECORDS
END STORE DB-RECORD

*** DATE AND TIME 2017-06-28  11:26:39
END CLOSE
*** DATE AND TIME 2017-06-28  11:26:39

***** DIAGNOSTIC SUMMARY OF BINILOAD

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

             55 RECORDS  STORED

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          184
***** NORMAL END   BINILOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:39
```

BINILOAD Eingabedatei

Beispiel zum Zusammenwirken von BINILOAD und BOUTLOAD siehe "Beispiele".

5.2 Sätze kopieren, löschen und entladen mit BOUTLOAD

Mit BOUTLOAD können Sie sehr schnell Satzarten einer Datenbank kopieren, löschen und entladen.

Sie können BOUTLOAD verwenden, wenn Sie eine Datenbank ganz oder teilweise entladen und neu laden wollen oder wenn Sie Auswertungen der Daten machen wollen.
Zum Umstrukturieren mit BALTER ist ein Entladen von Satzarten in wenigen Ausnahmefällen erforderlich oder zu empfehlen, z.B. bei der Umwandlung in DUPLICATES NOT ALLOWED, um Duplikate zu erkennen.

Die Dateien, die BOUTLOAD erzeugt, können von BINILOAD wieder gelesen werden.
Außerdem gibt BOUTLOAD die Steueranweisungen für einen folgenden BINILOAD-Lauf auf SYSLST aus, wenn der Parameter SET-INFORMATION auf YES gesetzt ist. Wenn Sie SYSLST einer Datei zuweisen, können Sie diese anschließend ändern und für BINILOAD benutzen.

BOUTLOAD kann auch Sätze mit Feldern variabler Länge entladen.

5.2.1 Beschreibung der Funktionen

Mit BOUTLOAD können Sie folgende Funktionen ausführen:

- Satzarten kopieren aus einer Datenbank in Ausgabedateien
- Satzarten löschen aus einer Datenbank
- Satzarten entladen aus einer Datenbank in Ausgabedateien

Sie können außerdem alle Sätze einer Satzart, die sich in einem bestimmten Realm befinden, in eine Ausgabedatei kopieren.

Sie können den Inhalt einer ganzen Datenbank löschen und damit die Datenbank neu formatieren.

BOUTLOAD legt beim Kopieren und Entladen die Sätze der angegebenen Satzarten in einer Ausgabedatei pro Satzart ab. Diese Ausgabedateien eignen sich auch als Eingabedateien für BINILOAD.

BOUTLOAD kopiert oder löscht die Sätze der angegebenen Satzarten in einem einzigen Durchgang durch die Datenbank; d.h. es behandelt mehrere Satzarten gleichzeitig und füllt auch die Ausgabedateien gleichzeitig. Die Konsistenz der Datenbank bleibt in jedem Fall erhalten.

Das Kopieren der Satzarten mit Set-Information-Ausgabe ist auch parallel zu einem DBH mit RETRIEVAL-Zugriff oder bei einer Schattendatenbank möglich.

Das Kopieren ohne Set-Information-Ausgabe ist auch parallel zum DBH erlaubt, sofern die Datenbank eingehängt ist und seit dem Einhängen und bis zum Abschluss des BOUTLOAD-Laufes nicht geändert wird. Das Einhalten dieser Bedingung müssen Sie mit geeigneten organisatorischen Maßnahmen sicherstellen; dies wird von BOUTLOAD nicht geprüft.

Satzarten kopieren (COPY-RECORD)

Sie können in einem BOUTLOAD-Lauf eine, mehrere oder alle Satzarten der Datenbank kopieren.

Falls Sie keine Ausgabedateien eingerichtet haben, legt BOUTLOAD diese Ausgabedateien an: pro Satzart eine Ausgabedatei.

Die Datenbank bleibt unverändert.

BOUTLOAD liest die Sätze in der physischen Reihenfolge, wie sie in der Datenbank vorliegen.

BOUTLOAD kopiert den Benutzerteil des Datenbanksatzes. Komprimierte Sätze werden dekomprimiert; Sätze mit variablen Feldern werden auf maximale Länge mit Leerzeichen aufgefüllt, wobei das Längenfeld des variablen Feldes erhalten bleibt. Eine von BOUTLOAD erzeugte Datei mit variablen Feldern kann von BINILOAD nicht verarbeitet werden.

Den Aufbau des Ausgabesatzes finden Sie im Abschnitt "Aufbau des Ausgabesatzes" ("Ausgabedateien und BOUTLOAD-Lauf vorbereiten").

Das Kopieren mit SET-INFORMATION=NO ist auch für eine inkonsistente Datenbank erlaubt.

Sätze einer Satzart aus einem Realm kopieren (COPY-RECORD, REALM-NAME)

Sie können in einem BOUTLOAD-Lauf auch nur die Sätze von einer oder mehreren Satzarten aus einem Realm kopieren. In diesem Fall wird die Area-Reference nicht im Ausgabesatz abgelegt.

Falls Sie keine Ausgabedateien eingerichtet haben, legt BOUTLOAD sie an: pro Satzart eine Ausgabedatei.

Der Realm bleibt unverändert.

> ¡ Bei Sets mit ORDER IS FIRST/LAST/NEXT/PRIOR/IMMATERIAL kann sich beim Wieder-Speichern die Sortierfolge der Sätze in den Set Occurrences ändern.

Satzarten löschen (REMOVE-RECORD)

Sie können in einem BOUTLOAD-Lauf eine, mehrere, oder alle Satzarten der Datenbank löschen. Es werden die Satzarten mit allen Verweisen auf die Sätze in zugehörigen Tabellen, in Ownersätzen und in DBTT-Einträgen gelöscht.

BOUTLOAD löscht mehrere Satzarten gleichzeitig in einem Durchgang durch die Datenbank, wobei Sie die Hierarchie in der Datenbank (Member-Owner-Beziehungen) berücksichtigen müssen: Die Membersatzarten müssen vor oder zusammen mit den Ownersatzarten gelöscht werden.

> ¡ Bei der Funktion REMOVE-RECORD ist AFIM-Logging nur bei Angabe einzelner Satzarten erlaubt, nicht jedoch bei der Angabe REMOVE-RECORD *ALL. Wenn Sie REMOVE-RECORD *ALL angeben, müssen Sie zuvor gegebenenfalls das AFIM-Logging mit BMEND abschalten.

Satzarten entladen (EXPORT-RECORD)

Sie können in einem BOUTLOAD-Lauf eine, mehrere oder alle Satzarten der Datenbank entladen.

Diese Funktion ist eine Kombination aus Kopieren und Löschen. Sie müssen genauso wie beim Löschen die Hierarchie in der Datenbank beachten.

Bei Satzarten, deren Sätze auf mehrere Realms verteilt sind, ist das Entladen von Sätzen aus nur einem Realm nicht möglich.

Beim Entladen aller Satzarten wird ebenso wie beim Löschen aller Satzarten die Datenbank neu formatiert.

> ¡ Bei der Funktion EXPORT-RECORD ist AFIM-Logging nur bei Angabe einzelner Satzarten erlaubt, nicht jedoch bei der Angabe EXPORT-RECORD *ALL. Wenn Sie EXPORT-RECORD *ALL angeben, müssen Sie zuvor gegebenenfalls das AFIM-Logging mit BMEND abschalten.

Alle Satzarten einer Datenbank löschen und entladen (REMOVE/EXPORT-RECORD,RECORD-NAME=*ALL)

Sie können in einem BOUTLOAD-Lauf alle Satzarten einer Datenbank löschen bzw. entladen. Dabei wird die Datenbank neu formatiert. Die Schema- und Subschemastruktur der Datenbank bleibt erhalten. Die FPA-Seiten, DBTTs, CALC-Seiten und Ankersätze werden zurückverlagert (siehe BFORMAT, "Benutzerrealms formatieren mit BFORMAT").
Für einen Formatierungslauf müssen alle Realms der Datenbank zur Verfügung stehen.

Das Formatieren mit REMOVE-RECORD,RECORD-NAME=*ALL ist auch für eine inkonsistente Datenbank erlaubt (z.B. nach Abbruch eines BOUTLOAD-Laufs).

> ¡ Bei den Funktionen REMOVE-RECORD und EXPORT-RECORD ist AFIM-Logging nur bei Angabe einzelner Satzarten erlaubt, nicht jedoch bei der Angabe REMOVE-RECORD *ALL bzw. EXPORT-RECORD *ALL.
> Wenn Sie REMOVE-RECORD *ALL oder EXPORT-RECORD *ALL angeben, müssen Sie zuvor gegebenenfalls das AFIM-Logging mit BMEND abschalten.

Die aktuelle Einstellung der Online DBTT-Erweiterbarkeit der Satzarten bleibt erhalten.

Zugriffsrechte bei den einzelnen Funktionen

	nur Konfigurationskennung	RETRIEVAL	EXCLUSIVE
kopieren	-	x	-
löschen	x	-	x
entladen	x	-	x

Tabelle 25: Zugriffsrechte bei den einzelnen Funktionen

i Soll eine Datenbank komplett entladen werden, empfiehlt es sich aus Laufzeitgründen zunächst mehrere BOUTLOAD-Läufe mit der Funktion Kopieren parallel ablaufen zu lassen und danach einen BOUTLOAD-Lauf mit der Funktion Löschen durchzuführen.

5.2.2 Ausgabedateien und BOUTLOAD-Lauf vorbereiten

Die einzelnen Ausgabedateien für BOUTLOAD können Sie mit folgenden Kommandos einrichten:

```
/CREATE-FILE FILE-NAME=dbname.RECnnnnn[.mmmmm] [,SUPPORT= ...]
/ADD-FILE-LINK LINK-NAME=linkname,FILE-NAME=dbname.RECnnnnn[.mmmmm]
    [,BUFFER-LENGTH=xxx][,FILE-SEQUENCE=*NEW]
```

dbname

Name der Datenbank, die behandelt wird

nnnnn

Nummer der Rec-Ref; fünfstellig mit führenden Nullen

mmmmm

Nummer der Area-Ref; fünfstellig mit führenden Nullen;
diese Angabe ist erforderlich, wenn Sie Sätze aus nur einem Realm kopieren wollen.

SUPPORT

Mit der SPACE-Angabe des SUPPORT-Operanden können Sie die Speicherplatzgröße festlegen. Dies ist nur bei Platte zulässig.

linkname

Einen frei wählbaren Linknamen müssen Sie angeben bei Angabe von FILE-SEQUENCE=*NEW, oder bei Änderung der BUFFER-LENGTH. Zur Ausgabedatei
darf dann nur ein TFT-Eintrag existieren.

xxx

Voreinstellung

- bei Platte: *STD(SIZE=4)
- bei Band:
  die BUFFER-LENGTH richtet sich nach der Länge der Sätze:
  mindestens vier PAM-Seiten. Der Wert wird auf ein ganzzahliges Vielfaches der Satzlänge und ein ganzzahliges Vielfaches eines Doppelwortes aufgerundet.

Bei Ausgabe auf Band sollten Sie die BUFFER-LENGTH mit einer direkten Zahlenangabe angeben, nicht Standardblock, da die Angabe von Standardblöcken STD die Ausgabedatei vergrößert.

FILE-SEQUENCE=*NEW

nur bei Band zulässig, wenn bei mehreren BOUTLOAD-Läufen auf die gleiche Bandmenge zugegriffen werden soll.

Die auszugebende Datenmenge errechnen Sie so:

Anzahl der Sätze x *Satzlänge* Bytes

Die Länge der Sätze errechnen Sie so:

- für Sätze mit Setinformation in einer 2-Kbyte-Datenbank:

```
Satzlänge = Satzlänge nach SIA-Protokoll - Länge der Systeminformation
          + 4 * (Anzahl der nicht-singulären Sets, in denen der Satz
                 Member ist + 1)
          + 1 * (Anzahl der singulären Sets, in denen der Satz Member
                 ist, außer bei MANDATORY AUTOMATIC Membern)
```

- für Sätze mit Setinformation in einer 4-Kbyte-/8-Kbyte-Datenbank:

```
Satzlänge = Satzlänge nach SIA-Protokoll - Länge der Systeminformation
          + 8 * (Anzahl der nicht-singulären Sets, in denen der Satz
                 Member ist + 1)
          + 1 * (Anzahl der singulären Sets, in denen der Satz Member
                 ist, außer bei MANDATORY AUTOMATIC Membern)
```

- für Sätze ohne Setinformation:

```
Satzlänge = Satzlänge nach SIA-Protokoll - Länge der Systeminformation
```

Bei auf Realms verteilten Satzarten kommen zur Satzlänge noch fünf Bytes für die Area-Ref hinzu, wenn die Sätze aus mehreren Realms kopiert oder extrahiert werden.

Die Sätze werden nämlich immer in eine Ausgabedatei pro Satzart kopiert, wichtig für das Auftreten der Area-Reference ist also die Herkunft aus mehr als einem Realm.

Die Anzahl der Sätze können Sie mit dem Dienstprogramm BSTATUS ermitteln.

Wenn Sie die Ausgabedateien nicht vorher eingerichtet haben, legt BOUTLOAD sie auf eine gemeinschaftliche Platte. Die Größe der jeweiligen Datei errechnet BOUTLOAD aus der max. Anzahl der DBTT-Einträge der betroffenen Satzart.

> **i** Bei Ausgabedateien auf Bändern müssen Sie darauf achten, dass so viel Bandgeräte wie Satzarten, die gleichzeitig entladen werden sollen, zur Verfügung stehen, weil BOUTLOAD die Satzarten gleichzeitig kopiert.

CSV-Ausgabedatei

Es ist nicht zwingend erforderlich, die CSV-Ausgabedatei zu erstellen. Sie wird stets vom Hilfsprogramm BOUTLOAD auf einer gemeinschaftlichen Platte erstellt.

Der Name der CSV-Ausgabedatei besteht aus dem Dateinamen der Ausgabedatei und dem Suffix „CSV“:

dbname.REC*nnnnn*[.*mmmmm*].CSV

dbname

Name der zu verarbeitenden Datenbank.

nnnnn

5-stellige Satzreferenznummer mit führenden Nullen.

mmmmm

5-stellige Area-Referenznummer mit führenden Nullen. Diese Spezifikation ist erforderlich, wenn Sätze nur aus einem Realm kopiert werden.

CSV

Suffix für die CSV-Ausgabedatei.

Die Sätze werden stets in eine CSV-Ausgabedatei pro Satzart kopiert.

Wenn CSV-OUTPUT = *YES angegeben ist, muss der DBCOM verfügbar sein.

Aufbau des Ausgabesatzes

Wenn BOUTLOAD wegen der Anweisung SET-INFORMATION=YES die Set-Information mit ausgegeben hat, wird der Ausgabesatz in folgender Struktur angelegt:

```
| Database Key| Feld| Database Keys aller Owner | Benutzerteil| Area-Ref|
--------------+-----+---------------------------+-------------+----------
```

- der Database Key des Satzes
- ein ein Byte langes Feld mit dem Inhalt
  X’00’ = Member eingefügt
  X’FF’ = Member nicht eingefügt
  (für alle singulären Sets, in denen der Satz Member ist, außer bei MANDATORY AUTOMATIC Membern)
- die Database Keys der Owner der nicht singulären Sets, in denen der Satz Member ist
- Ist der Satz nicht im Set eingehängt, wird der Database Key des Owners auf High-Value gesetzt (X'FFFFFFFF' bei einer 2-KB-Datenbank, bzw. X'FFFFFFFFFFFFFFFF' bei einer 4/8-KB-Datenbank)
- Benutzerteil
- die Area-Reference (Realm-Referenz) in der Länge von fünf Byte bei auf Realms verteilten Satzarten, wenn deren Sätze aus mehreren Realms kopiert werden. Die Sätze werden nämlich immer in eine Ausgabedatei pro Satzart kopiert, wichtig für das Auftreten der Area-Reference ist also die Herkunft aus mehr als einem Realm.

Wenn BOUTLOAD Set-Informationen zu den einzelnen Sätzen ausgibt, ist im BOUTLOAD-Protokoll, das die Anweisungen für einen nachfolgenden BINILOAD-Lauf enthält, die Länge der Database-Key-Werte angegeben (Länge „4“ bei einer 2-Kbyte-Datenbank, Länge „8“ bei einer 4-Kbyte-/8-Kbyte-Datenbank).

Ohne Set-Information besteht der Ausgabesatz nur aus dem Benutzerteil.

CSV-Ausgabedaten

Das folgende Beispiel zeigt einen Teil einer Datenausgabe in eine CSV-Ausgabedatei in einem mit Microsoft EXCEL vergleichbaren Präsentationsmodus. Bei einer CSV-Datei sind die einzelnen Werte dagegen durch ein Semikolon (;) voneinander getrennt.

BOUTLOAD	CSV V1.20	28.03.2014	14:38:53									
DB-NAME	DATABASE NAME	DB1										
INFO 01	RECORD NAME	RECORD2										
INFO 02	RECORD REF	2										
INFO 03	REALM NAME	AREA1										
INFO 04	REALM REF	3										
FIELDS	DB Key Ref	DB Key RSQ	Member SYS-1	Owner DB Key Ref S1	Owner DB Key RSQ S1	R1	R2-1 (1,1)	R2-1 (1,2)	R2-2 (1,1,1)	R2-2 (1,1,2)	...	Area-ref.
RE-CORD	2	1	Y	1	6	15,7	Y	A	YZ	AC	...	3
RE-CORD	2	3	N	1	3	-47,1	A	B	BC	AB	...	3

Die Kopfzeile der Datenausgabe enthält bis zu 6 Zeilen, wenn ein Realm-Name in COPY-RECORD angegeben wurde:

- Die erste Zeile enthält den Namen der Hilfsprogrammroutine, die jeweilige Ausgabeformatversion der Hilfsprogrammroutine sowie das Datum und die Uhrzeit der CSV-Ausgabeerstellung.
- Die nächsten Zeilen enthalten den Datenbanknamen, den Satznamen und die Satzreferenz. Wenn ein Realm-Name angegeben wurde, enthält die Kopfzeile auch den Realm-Namen und die Realm-Referenz.

Die Kopfzeile der Satzausgabe kann folgende Felder enthalten:

DB Key Ref, DB Key RSQ

Feldnamen für den Datenbankschlüssel des Satzes

Member *set-name*

Feldname für ein Feld mit einer Länge von einem Byte mit folgendem Inhalt:

Y Member in den SYSTEM-Set *set-name* eingefügt

N Member nicht in den SYSTEM-Set *set-name* eingefügt

(für alle Einzel-Sets, in denen der Satz ein Member ist, außer für MANDATORY, AUTOMATIC Member)

Owner DB Key Ref *set-name*, Owner DB Key RSQ *set-name*

Wenn der Satz ein Membersatz ist, werden Datenbankschlüssel aller Owner zusätzlich ausgegeben.

Die Feldnamen nach dem Benutzerschema

Area ref

Im Falle einer Satzart, die in Realms verteilt wird, wenn deren Sätze aus mehreren Realms kopiert werden

Wenn ein Feld Teil einer Wiederholungsgruppe oder eines Vektors ist, wird der Indexwert an den Feldnamen dieses Felds angefügt.

Felder des Typs DBKEY werden im folgenden Format ausgegeben:
DB KEY REF und DB KEY RSQ.

BOUTLOAD konvertiert binäre und numerische Daten wie folgt in ein druckbares Format:

- Das Dezimalzeichen wird durch das Zeichen "Komma" (",") dargestellt.
- Alphanumerische Felder mit variabler Länge werden nach der aktuellen Länge des variablen Elements ausgegeben.
- Felder in einem nationalen Typ werden in einen benutzerdefinierten Zeichensatz konvertiert, wenn dies möglich ist.

Um einen benutzerdefinierten Standard-Zeichensatz zuzuweisen oder abzurufen, gehen Sie wie folgt vor:

> Um einen benutzerdefinierten Standard-Zeichensatz zuzuweisen, verwenden Sie die Befehle ADD-USER oder MODIFY-USER.

> Um den benutzerdefinierten Standard-Zeichensatz abzurufen, verwenden Sie den Befehl SHOW-USER-ATTRIBUTES.

Für die Konvertierung von einem nationalen Datentyp muss das XHCS-Subsystem im System verfügbar sein. Wenn Zeichen des nationalen Datentyps wegen eines Fehlers des XHCS-Subsystems nicht konvertiert werden können, wird die Warnung 3935 ausgegeben, die Ausgabe in die CSV-Datei wird beendet, und die CSV-Datei wird gelöscht.

`3935 NATIONAL CHARACTERS CANNOT BE CONVERTED: XHCS RETURN CODE: `*`returncode`*

Wenn eine Satzart einen nationalen Datentyp enthält, wird die entsprechende CSV-Datei mit dem CODED-CHARACTER-SET der USER-ID angelegt.
Wenn das CODED-CHARACTER-SET des Users nicht bestimmt werden kann, wird die Warnung 3936 ausgegebe, die CSV-Ausgabe wird für diese Satzart beendet, und die CSV-Datei wird gelöscht.

`3936 USER CODED CHARACTER SET CANNOT BE DETERMINED: SRMUINFI RETURN CODE: `*`returncode`*

Wenn ein nationales Zeichen nicht in einen benutzerdefinierten Zeichensatz konvertiert werden kann (weil keine Entsprechung vorhanden ist), wird dieses nationale Zeichen als Zeichen "Punkt" (".") ausgegeben. Zusätzlich wird eine Warnmeldung ausgegeben:

`3932 STRING CONVERSION WITH SUBSTITUTION BY DEFAULT CHARACTERS PERFORMED FOR RECORD `*`recordname`*`.`

Ein Semikolon (";") wird verwendet, um die einzelnen Werte voneinander zu trennen.

Alphanumerische Werte können bestimmte Zeichen wie Trennlinien/Trennzeichen (";"), neue Zeilen oder doppelte Anführungszeichen enthalten, die in verschiedenen Systemumgebungen besondere Bedeutungen haben. Alle alphanumerischen Werte werden mit doppelten Anführungszeichen umschlossen, damit diese Werte in anderen Systemumgebungen korrekt verarbeitet werden können. Wenn ein Wert eingebettete (doppelte) Anführungszeichen enthält, werden diese doppelten Anführungszeichen als zwei (doppelte) Anführungszeichen dargestellt.

5.2.3 BOUTLOAD-Protokoll für das Format des Ausgabesatzes

BOUTLOAD erzeugt ein Protokoll auf SYSLST, wenn der Parameter SET-INFORMATION auf YES gesetzt ist. Weisen Sie SYSLST einer Datei zu, können Sie die Datei als Eingabedatei für BINILOAD anpassen und die BINILOAD-Steueranweisungen verwenden. Siehe auch Beispiele ab "Beispiele".

Beispiel

```
BINILOAD PARAMETERS FOR RECORD : FARBEN                          , REC-REF :   11
SCHEMA NAME IS ARTIKELVERSAND
SUBSCHEMA NAME IS
USER FILE RECORD LENGTH IS   30
USER FILE BUFFER LENGTH IS 8192
INPUT FILE 'VERSAND.REC00011.00011'
INPUT RECORDNUMBER IS       25
STORE RECORD NAME IS FARBEN
RECORD-DBKEY IS DISPL IS 0 , LENGTH IS 8
RECORD-DISPL IS 0 , DISPL IS    8 , LENGTH IS   22
RECORD-AREA NAME IS ARTIKELRLM
END
```

Wenn BOUTLOAD ein Protokoll in CSV erzeugt, dann werden die notwendigen Anweisungen für das weitere Hochladen durch BINILOAD aus der CSV-Datei erzeugt und ebenfalls nach SYSLST ausgegeben, und zwar nach den üblichen Anweisungen für BINILOAD.

Zum Beispiel bedeutet “FORMAT: CSV” in der Zeile "BINILOAD PARAMETERS FOR RECORD", dass die folgenden Anweisungen für das Laden aus der CSV-Datei durch BINILOAD passend sind.

Beispiel

```
BINILOAD PARAMETERS FOR RECORD : ARTICLE , REC-REF : 9, FORMAT: CSV
SCHEMA NAME IS MAIL-ORDERS
SUBSCHEMA NAME IS
INPUT FILE 'SHIPPING.REC00009.00005.CSV ' FORMAT IS CSV
INPUT RECORDNUMBER IS 55
STORE RECORD NAME IS ARTICLE
INSERT INTO SET NAME IS P-ORD-SPEC
INSERT INTO SET NAME IS MIN-STOCK-LEVEL
INSERT INTO SET NAME IS ARTICLES-AVAILABLE
END
```

5.2.4 Systemumgebung von BOUTLOAD

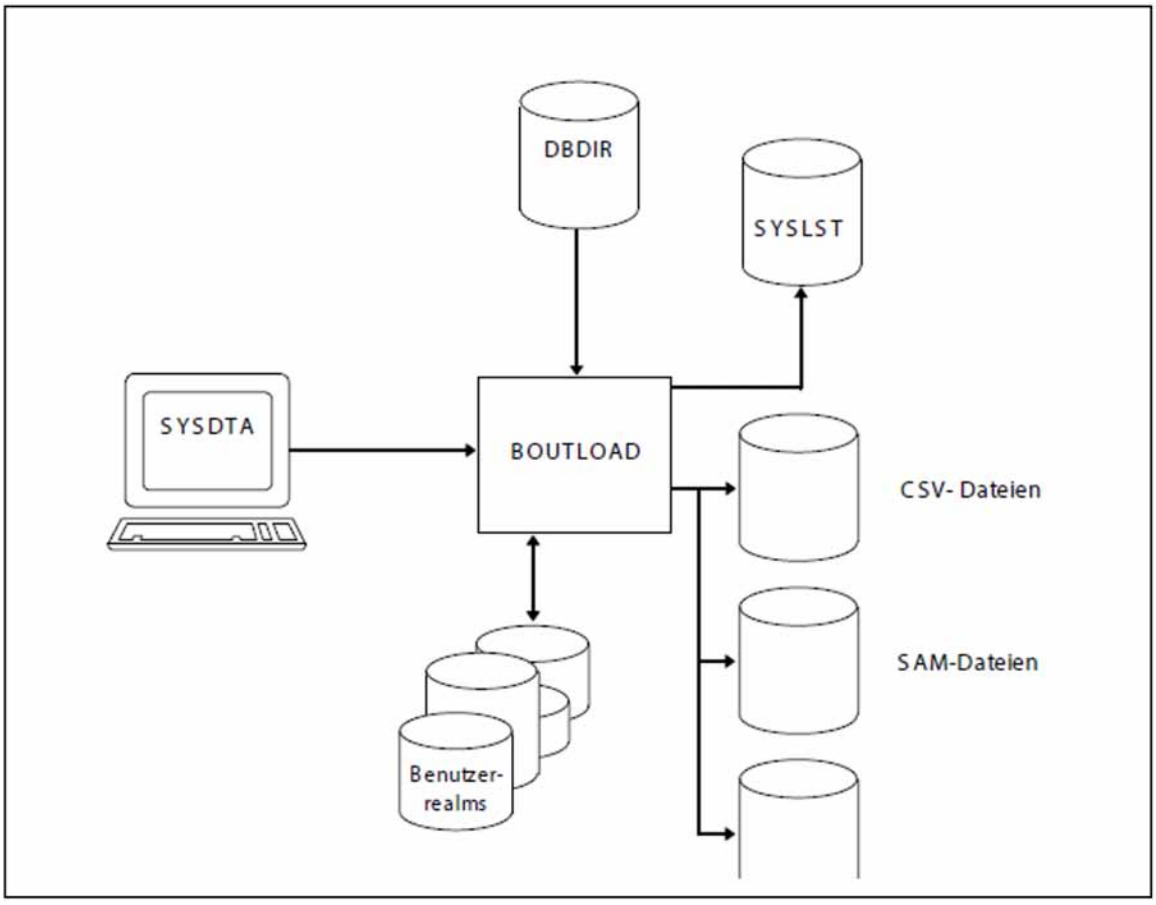

Bild 21: Systemumgebung von BOUTLOAD

In dieser Beschreibung wird davon ausgegangen, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

5.2.5 Anweisungen für BOUTLOAD

Die Anweisungsformate des Dienstprogramms BOUTLOAD entsprechen den Regeln von SDF (System Dialog Facility, siehe Handbücher „Dialogschnittstelle SDF“ und „Kommandos“)

Die bei den SDF-Formaten verwendeten Datentypen finden Sie in der Tabelle 5 ("SDF-Syntaxdarstellung").

Übersicht über die Anweisungen des BOUTLOAD

Anweisung	Bedeutung
COPY-RECORD **RECORD-NAME** = ***ALL** / list-poss(20): <recordname> / ***ALL-EXCEPT(...)** **,REALM-NAME** = ***ALL** / <realmname> **,SET-INFORMATION** = **YES** / **NO** **,CSV-OUTPUT** = ***NO** / ***YES (...)**	Alle Sätze der angegebenen Satzarten in die Ausgabedateien kopieren
END	BOUTLOAD beenden
EXPORT-RECORD **RECORD-NAME** = ***ALL** / list-poss(20): <recordname> / ***ALL-EXCEPT(...)** **,SET-INFORMATION** = **YES** / **NO**	Alle Sätz der angegebenen Satzarten in die Ausgabedateien entladen
OPEN-DATABASE **DATABASE-NAME** = <dbname> **,COPY-NAME** = ***NONE** / <copyname> **,USER-IDENTIFICATION** = ***OWN** / <userid>	Datenbank zuweisen
REMOVE-RECORD **RECORD-NAME** = ***ALL** / list-poss(20): <recordname> / ***ALL-EXCEPT(...)**	Alle Sätze der angegebenen Satzarten löschen

Tabelle 26: Anweisungen des BOUTLOAD

5.2.5.1 COPY-RECORD (Sätze in Ausgabedateien kopieren)

Mit dieser Anweisung kopieren Sie alle Sätze der angegebenen Satzarten in Ausgabedateien. Es ist auch möglich, die Ausgabedateien im CSV-Format auszugeben. Database-Key-Werte werden in der Form ausgegeben, in der sie in der Datenbank stehen, d.h. BOUTLOAD konvertiert die Database-Key-Werte bei der Ausgabe nicht von der kurzen in die lange Form und umgekehrt. Die Datenbank bleibt unverändert.

COPY-RECORD

RECORD-NAME = ***ALL** / list-poss(20): <recordname>/ ***ALL-EXCEPT(...)**

***ALL-EXCEPT(...)**

| **EXCEPT-NAME=** list-poss(20):<recordname>

,**REALM-NAME** = ***ALL** / <realmname>

,**SET-INFORMATION** = **YES** / **NO**

,CSV-OUTPUT = *NO / ***YES (...)**

***YES(...)**

| **OUTPUT= *STD / *FULL / *CSV-COMPATIBLE**

RECORD-NAME = *ALL / list-poss(20): <recordname>/ *ALL-EXCEPT(...)

***ALL**
Bei dieser Angabe ist nur diese eine Funktion im BOUTLOAD-Lauf zugelassen; d.h. es muss die END-Anweisung folgen.
Alle Sätze aller Satzarten werden kopiert.

<recordname>
Alle Sätze der angegebenen Satzart(en) werden kopiert.

***ALL-EXCEPT(...)**
Bei dieser Angabe ist nur diese eine Funktion im BOUTLOAD-Lauf zugelassen; d.h. es muss die END-Anweisung folgen.
Alle Sätze mit Ausnahme der angegebenen Satzarten werden kopiert.

EXCEPT-NAME= list-poss(20): <recordname>
Namen der Satzarten, die nicht kopiert werden sollen.

REALM-NAME = *ALL / <realmname>

***ALL**
Alle Sätze der angegebenen Satzart werden aus allen Realms kopiert, in denen sie vorkommen können.

<realmname>
Ist ein einzelner Realm angegeben, werden nur die Sätze dieses Realm in Ausgabedateien kopiert.
Bei dieser Angabe ist nur diese eine Funktion im BOUTLOAD-Lauf zugelassen; d.h. es muss die END-Anweisung folgen.

SET-INFORMATION = YES / NO
Mit dieser Angabe bestimmen Sie, ob in der jeweiligen Ausgabedatei pro Satz Set-Informationen gespeichert werden und ob Anweisungen für BINILOAD auf SYSLST geschrieben werden.

Vor dem Kopiervorgang wird nicht überprüft, ob zu kopierende Sätze in der Datenbank vorhanden sind. Sind keine Sätze vorhanden, so wird die zugehörige Ausgabedatei entleert, wenn sie vom Anwender eingerichtet wurde, oder wieder gelöscht, wenn sie von BOUTLOAD eingerichtet wurde.

CSV-OUTPUT = *NO / *YES

***NO**
BOUTLOAD gibt die Dateien in Ausgabedateien aus, jedoch nicht im CSV-Format.

***YES (...)**
BOUTLOAD gibt die Daten zusätzlich im CSV-Format aus.

OUTPUT= *STD / *FULL / *CSV-COMPATIBLE

*STD und *FULL erzeugen einen vollständigen CSV Output. *CSV-COMPATIBLE erzeugt einen Output, der nur aus der Headerzeile (mit den Feldnamen, aber ohne FIELD) und den Datenzeilen (ohne RECORD) besteht.

5.2.5.2 END (BOUTLOAD-Lauf beenden)

Mit dieser Anweisung beenden Sie den BOUTLOAD-Lauf. Die Anweisung END müssen Sie als letzte Anweisung angeben.

END

Diese Anweisung hat keine Operanden.

5.2.5.3 EXPORT-RECORD (Sätze in Ausgabedateien entladen)

Mit dieser Anweisung entladen Sie alle Sätze der angegebenen Satzarten aus der Datenbank in Ausgabedateien. Database-Key-Werte werden in der Form angegeben, in der sie in der Datenbank stehen, d.h. BOUTLOAD konvertiert die Database-Key-Werte bei der Ausgabe nicht von der kurzen in die lange Form und umgekehrt.

Sie dürfen sie nur verwenden, wenn Sie BOUTLOAD in der Kennung geladen haben, in der die Datenbank katalogisiert ist.

```
EXPORT-RECORD

RECORD-NAME = *ALL / list-poss(20): <recordname>/ *ALL-EXCEPT(...)

    *ALL-EXCEPT(...)

        |    EXCEPT-NAME= list-poss(20):<recordname>

,SET-INFORMATION = YES / NO
```

RECORD-NAME = *ALL / list-poss(20): <recordname>/ *ALL-EXCEPT(...)

***ALL**
Bei dieser Angabe ist nur diese eine Funktion im BOUTLOAD-Lauf zugelassen; d.h. es muss die END-Anweisung folgen.
Die Datenbank wird neu formatiert; alle Realms müssen zur Verfügung stehen.

<recordname>
Alle Sätze der angegebenen Satzart werden kopiert und in der Datenbank gelöscht.

***ALL-EXCEPT(...)**
Bei dieser Angabe ist nur diese eine Funktion im BOUTLOAD-Lauf zugelassen; d.h. es muss die END-Anweisung folgen.
Alle Sätze mit Ausnahme der angegebenen Satzarten werden kopiert und in der Datenbank gelöscht.

EXCEPT-NAME= list-poss(20): <recordname>
Namen der Satzarten, die nicht kopiert und in der Datenbank gelöscht werden sollen.

SET-INFORMATION = YES / NO
Mit dieser Angabe bestimmen Sie, ob in der jeweiligen Ausgabedatei pro Satz Setinformationen gespeichert werden und ob Anweisungen für BINILOAD auf SYSLST geschrieben werden.

Wenn BOUTLOAD Set-Informationen zu den einzelnen Sätzen ausgibt, ist im BOUTLOAD-Protokoll, das die Anweisungen für einen nachfolgenden BINILOAD-Lauf enthält, die Länge der Database-Key-Werte angegeben (Länge „4“ bei einer 2-Kbyte-Datenbank, Länge „8“ bei einer 4-Kbyte-/8-Kbyte-Datenbank).

Das Kopieren von Satzarten mit SET-INFORMATION=NO ist auch für eine inkonsistente Datenbank erlaubt.

Vor dem Kopiervorgang wird nicht überprüft, ob zu kopierende Sätze in der Datenbank vorhanden sind. Sind keine Sätze vorhanden, so wird die zugehörige Ausgabedatei entleert, wenn sie vom Anwender eingerichtet wurde, oder wieder gelöscht, wenn sie von BOUTLOAD eingerichtet wurde.

Bei der Angabe der Satzarten müssen Sie die hierarchische Struktur im Schema der Datenbank beachten; d.h. Sie müssen die Membersatzarten vor oder zusammen mit den Ownersatzarten entladen. Sind die Angaben nicht korrekt, wird der BOUTLOAD-Lauf abgebrochen.

Wenn Sie nachfolgend Sätze in die Datenbank speichern, beginnt die Vergabe der DB-Keys wieder bei RSQ=1. Die DB-Keys sind nunmehr auch (einmalig) verwendbar, wenn sie mit der BMODTT-Anweisung KEEP für eine Wiederverwendung durch den DBH gesperrt sind. Ein zusätzlicher BMODTT-Lauf mit der Anweisung REMOVE ist nicht notwendig.

5.2.5.4 OPEN-DATABASE (Datenbank zuweisen)

Mit dieser Anweisung weisen Sie die Datenbank zu.
Sie müssen sie als erste Anweisung angeben, wenn Sie die Datenbank nicht zugewiesen haben mit:

`/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=`*`dbname`*`.DBDIR`

Wenn Sie das SET-FILE-LINK-Kommando angegeben haben, wird eine OPEN-DATABASE-Anweisung als fehlerhaft abgewiesen.

OPEN-DATABASE
DATABASE-NAME = <dbname> ,**COPYNAME** = ***NONE** / <copyname> ,**USER-IDENTIFICATION** = ***OWN**/ <userid>

DATABASE-NAME = <dbname>
Name der Datenbank. Ein Anwender kann nur eine Datenbank bearbeiten, die in seiner eigenen Kennung liegt. Eine Datenbank aus einer fremden Kennung kann nur von der TSOS-Kennung des Systemverwalters bearbeitet werden.

COPY-NAME = *NONE / <copyname>

> ***NONE**
> Die Originaldatenbank wird eröffnet.
>
> **<copyname>**
> Die Schattendatenbank wird eröffnet.

USER-IDENTIFICATION = *OWN / <userid>

> ***OWN**
> Die Datenbank liegt in der eigenen Kennung.
>
> **<userid>**
> Die Datenbank liegt unter einer fremden Kennung. Diese Angabe ist nur von der TSOS-Kennung aus erlaubt.

Der Linkname DATABASE bleibt solange gültig, bis er durch das Kommando REMOVE-FILE-LINK aufgehoben wird.

Die OPEN-DATABASE-Anweisung gilt nur bis zum Ende des BOUTLOAD-Laufs.

5.2.5.5 REMOVE-RECORD (Sätze löschen)

Mit dieser Anweisung löschen Sie alle Sätze der angegebenen Satzarten in der Datenbank.

Sie dürfen sie nur verwenden, wenn Sie BOUTLOAD in der Kennung geladen haben, in der die Datenbank katalogisiert ist.

REMOVE-RECORD

RECORD-NAME = ***ALL** / list-poss(20): <recordname>/ ***ALL-EXCEPT(...)**

***ALL-EXCEPT(...)**

| **EXCEPT-NAME=** list-poss(20):<recordname>

RECORD-NAME = *ALL / list-poss(20): <recordname>/ *ALL-EXCEPT(...)

***ALL**
Bei dieser Angabe ist nur diese eine Funktion im BOUTLOAD-Lauf zugelassen; d.h. es muss die END-Anweisung folgen.
Die Datenbank wird neu formatiert; alle Realms müssen zur Verfügung stehen.

Diese Anweisung ist auch für eine inkonsistente Datenbank zugelassen (System-Break-Bit gesetzt).

<recordname>
Alle Sätze der angegebenen Satzart werden in der Datenbank gelöscht.

***ALL-EXCEPT(...)**
Bei dieser Angabe ist nur diese eine Funktion im BOUTLOAD-Lauf zugelassen; d.h. es muss die END-Anweisung folgen.
Alle Sätze mit Ausnahme der angegebenen Satzarten werden in der Datenbank gelöscht.

EXCEPT-NAME= list-poss(20): <recordname>
Namen der Satzarten, die nicht gelöscht werden sollen.

Bei der Angabe der Satzarten müssen Sie die hierarchische Struktur im Schema der Datenbank beachten; d.h. Sie müssen die Membersatzarten vor oder zusammen mit den Ownersatzarten löschen. Sind die Angaben nicht korrekt, wird der BOUTLOAD-Lauf abgebrochen.

Wenn Sie nachfolgend Sätze in die Datenbank speichern, beginnt die Vergabe der DB-Keys wieder bei RSQ=1. Die DB-Keys sind nunmehr auch (einmalig) verwendbar, wenn sie mit der BMODTT-Anweisung KEEP für eine Wiederverwendung durch den DBH gesperrt sind. Ein zusätzlicher BMODTT-Lauf mit der Anweisung REMOVE ist nicht notwendig.

5.2.6 Kommandofolge zum Starten von BOUTLOAD

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
01 [/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR]
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /START-UDS-BOUTLOAD
04 [OPEN-DATABASE DATABASE-NAME=dbname]
05 boutload-Anweisungen
06 END
```

01,04 Sie müssen eine der beiden Zuweisungen für die Datenbank verwenden.

02 Die angegebene Version von BOUTLOAD wird ausgewählt.
Die Angabe der Version wird generell empfohlen, da mehrere UDS/SQL-Versionen parallel installiert sein können.

03 BOUTLOAD kann von jeder beliebigen Kennung aus aufgerufen werden. Das UDS/SQL-Dienstprogramm kann auch mit den Aliasnamen BOUTLOAD und START-UDS-OUTLOAD gestartet werden.

5.2.7 Beispiele

Beispiel zu BOUTLOAD

Es wird die Satzart FARBEN aus der Datenbank VERSAND kopiert.

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BOUTLOAD
***** START       BOUTLOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:37
//OPEN-DATABASE DATABASE-NAME=VERSAND
//COPY-RECORD RECORD-NAME=FARBEN,REALM-NAME=ARTIKELRLM
3903  AFTER " REALM-NAME = <NAME> " ONLY THE " END " STATEMENT IS ALLOWED
//END
***** INPUT CHECK SUCCESSFULLY TERMINATED
***** BEGIN SCAN OF USER-REALMS
***** REALM: ARTIKELRLM
***** SCAN OF USER-REALMS SUCCESSFULLY TERMINATED

***** DIAGNOSTIC SUMMARY OF BOUTLOAD

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          32
***** NORMAL END   BOUTLOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:37
```

Beispiel zu BOUTLOAD und BINILOAD

Kopieren und Löschen der Satzart MATERIALIEN mit BOUTLOAD und Speichern derselben Satzart mit BINILOAD. Dabei verwendet BINILOAD die von BOUTLOAD erzeugten Ausgaben für die Eingabe der Daten und für die Eingabe der BINILOAD-Anweisungen.

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BOUTLOAD
***** START       BOUTLOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:37
//OPEN-DATABASE DATABASE-NAME=VERSAND
//COPY-RECORD RECORD-NAME=MATERIALIEN,REALM-NAME=ARTIKELRLM
3903  AFTER " REALM-NAME = <NAME> " ONLY THE " END " STATEMENT IS ALLOWED
//END
***** INPUT CHECK SUCCESSFULLY TERMINATED
***** BEGIN SCAN OF USER-REALMS
***** REALM: ARTIKELRLM
***** SCAN OF USER-REALMS SUCCESSFULLY TERMINATED

BINILOAD PARAMETERS FOR RECORD : MATERIALIEN                  , REC-REF :   12
SCHEMA NAME IS ARTIKELVERSAND                                              - (1)
SUBSCHEMA NAME IS                                                          - (1)
USER FILE RECORD LENGTH IS   29                                            - (1)
USER FILE BUFFER LENGTH IS 8192                                            - (1)
INPUT FILE 'VERSAND.REC00012.00011'                                        - (1)
INPUT RECORDNUMBER IS      10                                              - (1)
STORE RECORD NAME IS MATERIALIEN                                           - (1)
RECORD-DBKEY IS DISPL IS 0 , LENGTH IS 8                                   - (1)
RECORD-DISPL IS 0 , DISPL IS    8 , LENGTH IS   21                         - (1)
RECORD-AREA NAME IS ARTIKELRLM                                             - (1)
END                                                                        - (1)

 ***** DIAGNOSTIC SUMMARY OF BOUTLOAD

              NO WARNINGS
              NO ERRORS
              NO SYSTEM-ERRORS

 ***** END OF DIAGNOSTIC SUMMARY
 ***** NR OF DATABASE ACCESSES  :          32
 ***** NORMAL END   BOUTLOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:37
.
.
.
```

(1) von BOUTLOAD erzeugte BINILOAD-Anweisungen, die als Eingabe für BINILOAD verwendet werden.

```
 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
 /START-UDS-BOUTLOAD
 ***** START       BOUTLOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:37
 //OPEN-DATABASE DATABASE-NAME=VERSAND
 //REMOVE-RECORD RECORD-NAME=MATERIALIEN
 //END
 ***** INPUT CHECK SUCCESSFULLY TERMINATED
 ***** NO OCCURRENCES OF MEMBER RECORDS DETECTED
 ***** BEGIN SCAN OF USER-REALMS
 ***** REALM: ARTIKELRLM
 ***** SCAN OF USER-REALMS SUCCESSFULLY TERMINATED

 ***** DIAGNOSTIC SUMMARY OF BOUTLOAD

              NO WARNINGS
              NO ERRORS
              NO SYSTEM-ERRORS

 ***** END OF DIAGNOSTIC SUMMARY
 ***** NR OF DATABASE ACCESSES  :          92
 ***** NORMAL END   BOUTLOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:38
.
.
.
```

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BINILOAD
***** START       BINILOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:39
EXECUTION WITH CHECK.                                                    - (1)
SCHEMA NAME IS ARTIKELVERSAND                                            - (1)
SUBSCHEMA NAME IS ADMIN                                                  - (1)
USER FILE RECORD LENGTH IS   29                                          - (1)
USER FILE BUFFER LENGTH IS 8192
INPUT FILE 'VERSAND.REC00012.00011'                                      - (1)
INPUT RECORDNUMBER IS      10                                            - (1)
STORE RECORD NAME IS MATERIALIEN                                         - (1)
RECORD-DBKEY IS DISPL IS 0 , LENGTH IS 8                                 - (1)
RECORD-DISPL IS 0 , DISPL IS    8 , LENGTH IS   21                       - (1)
RECORD-AREA NAME IS ARTIKELRLM                                           - (1)
END                                                                      - (1)
BEGIN CHECK-RUN
*** DATE AND TIME 2017-06-28  11:26:39
BEGIN ALLOCATION
*** DATE AND TIME 2017-06-28  11:26:39
SEARCHKEY ON RECORD-LEVEL:
--------------------------
REC REF:    12
RECORD NAME: MATERIALIEN
SEARCHKEY TABLE, DBTT_COLUMN_NR:     1
*** ICRELES: MOVE_ROUTINE SORTKF CREATED
BEGIN TABLE-PROCESSOR
*** DATE AND TIME 2017-06-28  11:26:40
SEARCHKEY ON RECORD-LEVEL:
--------------------------
REC REF:    12
RECORD NAME: MATERIALIEN
```

```
SEARCHKEY TABLE, DBTT_COLUMN_NR:     2
*** NO ERRORS DETECTED DURING CHECK-RUN
END CHECK-RUN
*** DATE AND TIME 2017-06-28  11:26:40
BEGIN ALLOCATION
*** DATE AND TIME 2017-06-28  11:26:40
*** DATABASE IS IN USE
SEARCHKEY ON RECORD-LEVEL:
--------------------------
REC REF:    12
RECORD NAME: MATERIALIEN
SEARCHKEY TABLE, DBTT_COLUMN_NR:     1
*** ICRELES: MOVE_ROUTINE SORTKF CREATED
BEGIN TABLE-PROCESSOR
*** DATE AND TIME 2017-06-28  11:26:40
SEARCHKEY ON RECORD-LEVEL:
--------------------------
REC REF:    12
RECORD NAME: MATERIALIEN
SEARCHKEY TABLE, DBTT_COLUMN_NR:     2
BEGIN STORE DB-RECORD
*** DATE AND TIME 2017-06-28  11:26:40
STORING DATABASE RECORDS
END STORE DB-RECORD
*** DATE AND TIME 2017-06-28  11:26:40
END CLOSE
*** DATE AND TIME 2017-06-28  11:26:40

***** DIAGNOSTIC SUMMARY OF BINILOAD

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

             10 RECORDS  STORED

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :           96
***** NORMAL END   BINILOAD     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:40
```

(1) von BOUTLOAD erzeugte BINILOAD-Anweisungen, die als Eingabe für BINILOAD verwendet werde

6 Datenbank umstrukturieren (BCHANGE, BALTER)

Umstrukturieren bedeutet, dass das Schema und die Speicherstruktur in einer Datenbank geändert werden, wenn Anwenderdaten bereits gespeichert sind.

Reine Umbenennungen, die sich nur auf das Schema auswirken, können Sie in einem Umbenennungszyklus vornehmen (siehe Kapitel „Datenbankobjekte umbenennen (BRENAME, BALTER)").

Die Aktivitäten, die beim Umstrukturieren erforderlich sind, gliedern sich in drei Abschnitte:

- vorbereitende Maßnahmen
- Umstrukturierungsprozess
- Folgeaktivitäten

Vorbereitende Maßnahmen

- Datenbankschema und Speicherstruktur analysieren und modifizieren
- Konsistenz der Datenbank prüfen
- Speicherplatz-Statistik analysieren
- ggf. After-Image-Logging mit BMEND ausschalten (siehe auch Abschnitt „Datenbanksichern")
- entweder
  - komplette Datenbank inklusive DBCOM, COSSD und HASHLIB vor dem Umstrukturierungsprozess sichern

  oder
  - HASHLIB, COSSD, DBDIR und DBCOM vor dem Umstrukturierungsprozess sichern
  - in einem Analyselauf mit den Anweisungen
    REPORT IS YES und EXECUTION IS NO
    ermitteln, welche Benutzerrealms benötigt werden
  - diese Benutzerrealms vor der Ausführungsphase des BALTER sichern

  Detaillierte Informationen zur Sicherung siehe Abschnitt „Datenbank sichern".

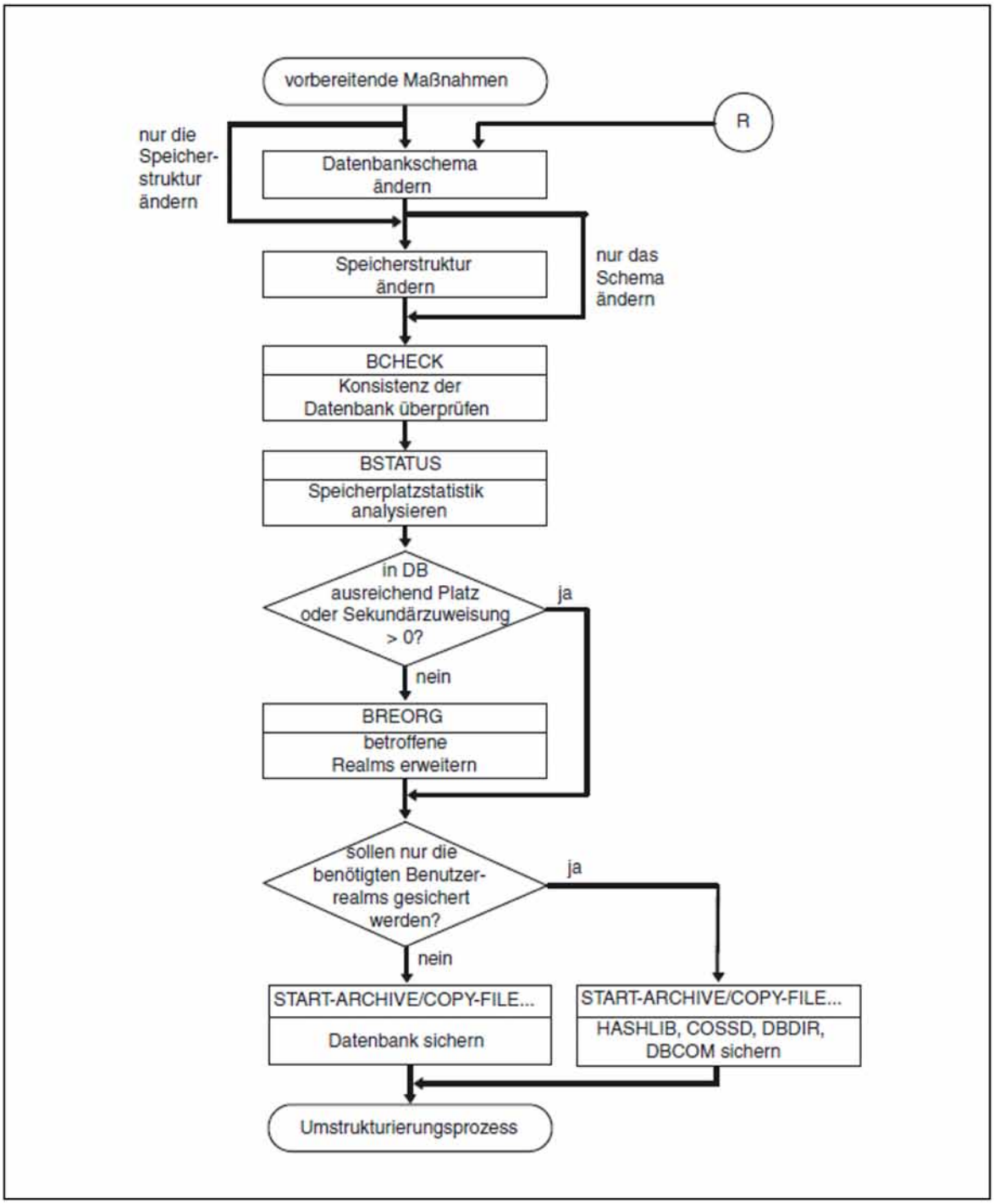

Bild 22: Vorbereitende Maßnahmen zum Umstrukturieren einer Datenbank

Umstrukturierungsprozess

Dies ist ein Ablauf, der dem Aufbauen einer Datenbank gleicht:

- mit BCHANGE bereiten Sie den DBDIR für die Aufnahme einer neuen SIA vor
- anschließend übersetzen Sie Ihre neuen DDL- und SSL-Definitionen und tragen die neue SIA im DBDIR ein
- BALTER passt schließlich den Datenbestand an das geänderte Schema an

i Der Umstrukturierungszyklus von BCHANGE/BALTER kann nicht mit dem Umbenennungszyklus von BRENAME/BALTER kombiniert werden. Eine Umbenennung in einem Umstrukturierungszyklus wird als Löschung des alten Feldes und Einfügung des neuen Feldes interpretiert. Dadurch kann es zu Datenverlust kommen.

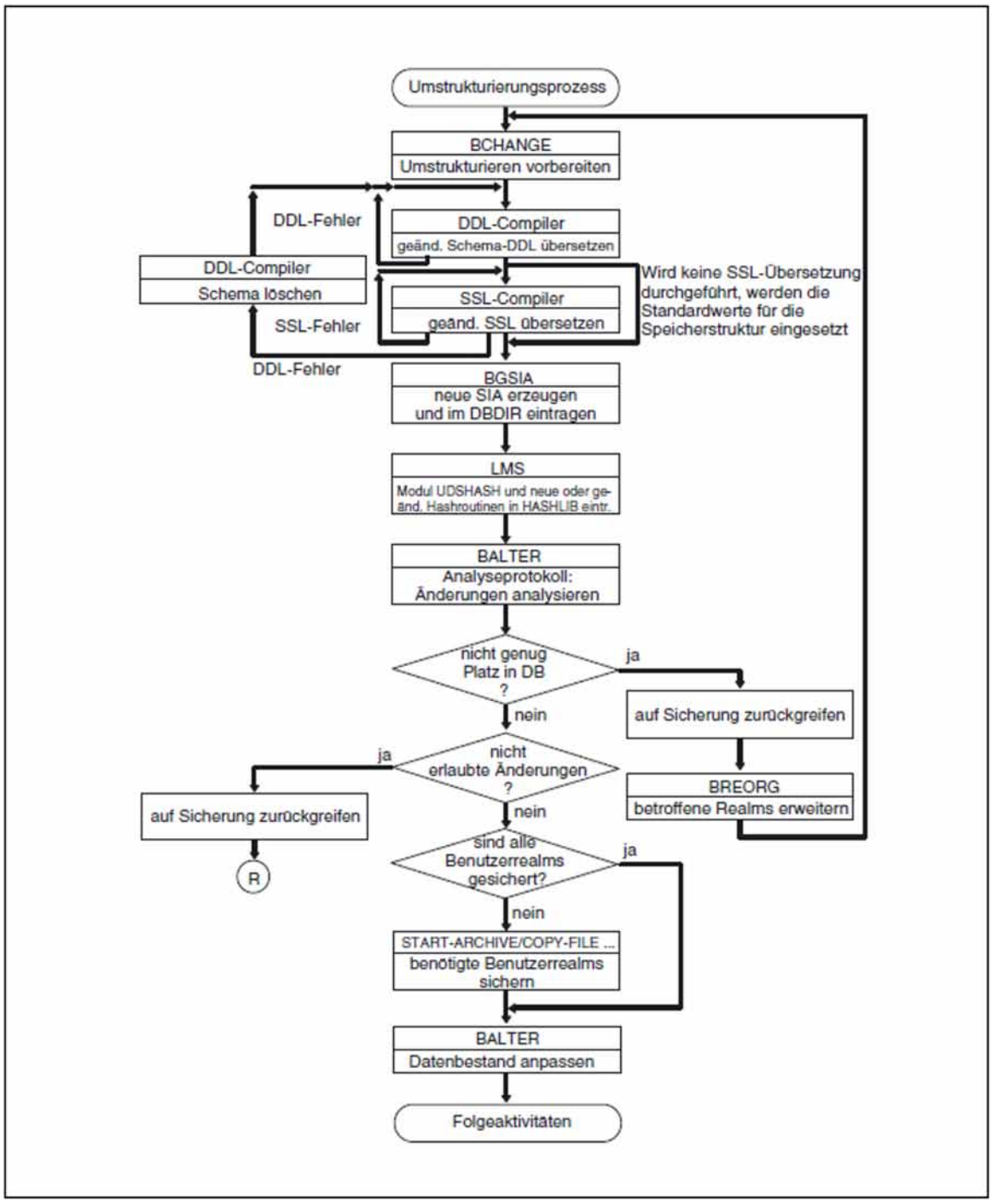

Bild 23: Umstrukturierungsprozess

Folgeaktivitäten

Nach dem Umstrukturieren müssen folgende Aktivitäten durchgeführt werden:

- Zugriffsberechtigungen neu eintragen, wenn in der Ausgangsdatenbank Benutzergruppennamen für Zugriffsrechte definiert sind
- Subschemata an das geänderte Schema anpassen
- DB-Anwenderprogramme an das neue Schema anpassen
- mit BREORG Sets und Hashbereiche reorganisieren.
- ggf. mit dem Dienstprogramm BMODTT die Wiederverwendung frei gewordener DB-Keys steuern (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BMODTT).

ℹ Durch den Umstrukturierungszyklus entsteht eine Logging-Lücke (siehe Handbuch „Datenbankbetrieb“, Langzeitsicherung). Deshalb müssen Sie nach dem Umstrukturierungszyklus durch Kopieren der geänderten Dateien zusammen mit den nicht geänderten Datenbankdateien eine neue Basis für die Langzeitsicherung aufbauen. Anschließend müssen Sie mit BMEND das After-Image-Logging wieder einschalten.

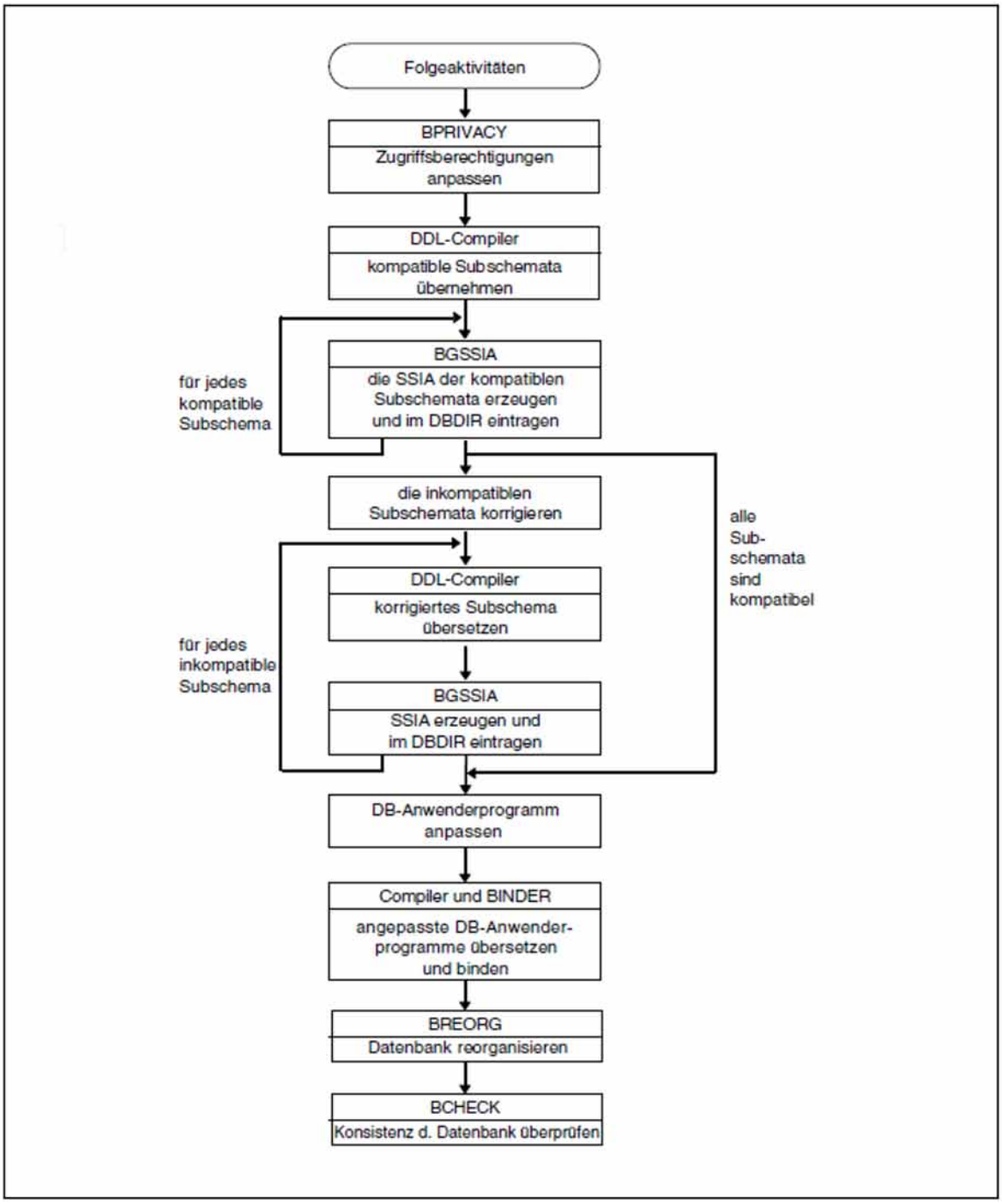

Bild 24: Aktivitäten nach dem Umstrukturieren der Datenbank

6.1 Schema-DDL ändern

Wollen Sie Ihre Schema-DDL ändern, so müssen Sie eine neue und vollständige Schema-Definition erstellen und diese neu übersetzen.

Mit BALTER können Sie folgende Schema-DDL-Änderungen bearbeiten:

- auf Realm-Ebene
  - Realms hinzufügen oder löschen
- auf Satzartebene
  - Satzarten hinzufügen oder löschen
  - den LOCATION MODE ändern
  - SEARCH-Keys neu definieren, löschen oder modifizieren
  - Felder hinzufügen, weglassen, umdefinieren, verlängern oder verkürzen
- auf Setebene
  - Sets hinzufügen oder löschen (mit Einschränkungen)
  - die ORDER-Klausel ändern (mit Einschränkungen)
  - die Sortierkriterien ändern
  - SEARCH-Keys neudefinieren, abschaffen oder modifizieren

Bei einer Umstrukturierung identifiziert BALTER die Datenbankelemente (Realms, Satzarten, Sets, Schlüssel, usw.) ausschließlich durch ihren Namen:

- BALTER erkennt Elemente als gleich, wenn sie vom gleichen Typ sind (z.B. Satzart) und ihre Namen in der alten und neuen Schema-DDL vorkommen.
- BALTER löscht Elemente, wenn für diesen Typ der Name in der neuen Schema-DDL nicht vorkommt.
- BALTER fügt Elemente hinzu, wenn für diesen Typ der Name in der alten Schema-DDL nicht vorkommt.

Dementsprechend können Sie Elemente nicht umbenennen oder im selben Umstrukturierungslauf ein Element löschen und ein Element gleichen Typs und gleichen Namens wieder hinzufügen. Zum Umbennenen von Feldern steht Ihnen das Dienstprogramm BRENAME zur Verfügung (siehe Kapitel „Datenbankobjekte umbenennen (BRENAME, BALTER)").

Bei dem BGSIA-Lauf während der Umstrukturierung bleiben die Nummern der Datenbankelemente erhalten; BGSIA ordnet die jeweiligen Nummern über den Elementnamen zu. Daher ist die Reihenfolge beliebig, in der Sie die Elemente in der neuen Schema-DDL definieren.

Die Klauseln der Schema-DDL und der SSL unterliegen bei einer Umstrukturierung den gleichen Regeln wie beim Definieren der Datenbank (siehe Handbuch „Entwerfen und Definieren“). Wenn Sie eine Klausel der Schema-DDL ändern, müssen Sie demnach alle damit zusammenhängenden Klauseln der Schema-DDL und der SSL so anpassen, dass sie diesen Regeln entsprechen.

Die in der Schema-DDL möglichen Änderungen und ihre Auswirkungen auf den Datenbestand sind im Folgenden ausführlich beschrieben.

6.1.1 Schema-Eintrag

```
SCHEMA NAME IS schemaname
```

schemaname

dürfen Sie wahlweise ändern

```
[PRIVACY LOCK FOR COPY IS literal-1[ OR literal-2]].
```

Die PRIVACY LOCK-Angaben können Sie beliebig verändern.

Auswirkung auf den Datenbestand

Der DDL-Compiler trägt die neuen PRIVACY LOCK-Angaben in den neuen DBCOM ein.

Ein Subschema, dessen PRIVACY KEY nicht mit den neuen PRIVACY LOCK-Angaben übereinstimmt, können Sie trotzdem weiterverwenden (siehe Abschnitt „Kompatible Subschemata übernehmen"). Die neuen PRIVACY LOCK-Angaben müssen Sie erst beim Übersetzen geänderter bzw. neuer Subschemata berücksichtigen.

6.1.2 Realm-Eintrag

```
AREA NAME IS realmname
```

Hinzufügen oder Löschen von Realms ist erlaubt.

Löschen Sie einen Realm, der Sätze einer Satzart enthält, die auch im neuen Schema noch vorhanden ist, so müssen Sie diese Sätze vor dem Umstrukturieren entladen, da BALTER keine Verlagerung von Sätzen in andere Realms vornimmt.
Einen gelöschten Realm hängt BALTER nur von der Datenbank ab; Sie müssen seine Datei nach Abschluss der Umstrukturierung mit dem ERASE-Kommando löschen.

Hinzugefügte Realms müssen Sie nicht mit BFORMAT formatieren; BALTER formatiert sie automatisch und hängt sie an die Datenbank an. Sie müssen sie aber vor dem ausführenden BALTER-Lauf per CREATE-FILE-Kommando einrichten.
Dabei ist darauf zu achten, dass die Sekundärzuweisung auf einen Wert größer 0 gesetzt wird, wenn der Realm online erweiterbar werden soll (siehe Handbuch „Datenbankbetrieb“, ACT INCR).

```
[AREA IS TEMPORARY].
```

Hinzufügen oder Löschen des Temporären Realms ist generell möglich.

Es ist nicht zulässig, einen Temporären Realm in einen nicht Temporären Realm und umgekehrt umzuwandeln.

6.1.3 Satz-Eintrag

```
RECORD NAME IS satzname
```

satzname

Satzarten dürfen Sie hinzufügen oder löschen, oder ihre Definition ändern.

Eine Satzart, die Sie löschen wollen, müssen Sie vorher nicht entladen; BALTER löscht alle zugehörigen Informationen wie Sätze, Hashbereiche, DBTT, und Tabellen, mit einer Ausnahme:
Komprimiert gespeicherte Sätze oder Sätze mit variablen Feldern müssen Sie vor der Umstrukturierung entladen, da BALTER sie nicht verarbeiten, also auch nicht löschen kann.

```
[LOCATION MODE IS {
 {DIRECT | DIRECT-LONG} {feldname-1 {IN | OF} satzname | bezeichner-1} |
  CALC[ hashroutine] USING feldname-2,... DUPLICATES ARE[ NOT] ALLOWED}]
```

Die LOCATION MODE-Klausel dürfen Sie beliebig verändern, hinzufügen oder weglassen.

CALC/DIRECT bzw. CALC/DIRECT-LONG

Umwandeln der Angabe DIRECT bzw. DIRECT-LONG in CALC und umgekehrt sowie das Weglassen oder Hinzufügen der LOCATION MODE-Klausel mit einer dieser Angaben ist zugelassen.

Auswirkung auf den Datenbestand

- DIRECT bzw. DIRECT-LONG -> CALC:
  BALTER legt einen neuen Hashbereich an und verlagert die Datensätze in den neu angelegten Hashbereich.
  Wenn die Satzart Membersatzart einer Liste ist, wird ein indirekter CALC angelegt. Bei einer verteilbaren Liste liegt der Hashbereich für alle Sätze in einem Realm.
- CALC -> DIRECT bzw. DIRECT-LONG:
  BALTER löscht alle für das Hashverfahren notwendigen Systeminformationen, verlagert aber nicht die bereits gespeicherten Sätze dieser Satzart in der Datenbank.
- Hinzufügen/Weglassen von LOCATION MODE IS CALC:
  Diese Änderung wirkt sich auf den Datenbestand genauso aus wie das Umwandeln der Angabe DIRECT bzw. DIRECT-LONG in CALC und umgekehrt.
- Hinzufügen/Weglassen von LOCATION MODE IS DIRECT/DIRECT-LONG:Diese Änderung wirkt sich auf den Datenbestand nicht aus; lediglich in den Anwenderprogrammen müssen Sie die Änderung berücksichtigen.

DIRECT bzw. DIRECT-LONG

feldname-1:
das Schlüsselfeld dürfen Sie nach Belieben ändern

bezeichner-1:
darf geändert werden

Auswirkung auf den Datenbestand

Auf den Datenbestand wirken sich diese Änderungen nicht aus;
lediglich in den Anwenderprogrammen müssen Sie die Änderung berücksichtigen.

CALC

hashroutine: dürfen Sie ändern;
dabei ist es erlaubt, von der Standard-Hashroutine von UDS/SQL zu einer eigenen Hashroutine zu wechseln oder umgekehrt, oder die eigene Hashroutine gegen eine neue eigene Hashroutine auszutauschen

feldname-2,..:
jede beliebige Änderung der Schlüsselfelder ist zugelassen

DUPLICATES...:
Duplikate dürfen Sie beliebig zulassen oder verbieten.

Auswirkung auf den Datenbestand

- Ändern Sie die Schlüsselfelder oder die Hashroutine, so richtet BALTER einen neuen Hashbereich ein und speichert die Datensätze um.
- Ändern Sie die Angabe DUPLICATES in NOT ALLOWED, so **müssen Sie beachten**, dass BALTER den Datenbestand auf vorhandene Duplikate nur in den Fällen prüft, in denen er die Schlüssel auf Grund weiterer Änderungen bearbeiten muss.
- Findet BALTER Datensätze mit doppelten Schlüsselwerten, so **eliminiert er sie nicht**. Verbieten Sie Duplikate, müssen Sie daher selber prüfen, ob Datensätze mit doppelten Schlüsselwerten vorkommen und wenn ja, den Datenbestand vor dem Umstrukturieren bereinigen.
- Doppelte Schlüsselwerte protokolliert BALTER nur in der EXECUTION-Phase, nicht im Analyseprotokoll; die Umstrukturierung setzt er anschließend fort.
- Diese Behandlung von Duplikaten gilt für alle Klauseln, die den Eintrag DUPLICATES enthalten!

```
WITHIN realmname-1[,realmname-2,... AREA-ID IS bezeichner-2]
```

realmname

Die Zuordnung von Satzarten zu bestimmten Realms dürfen Sie ändern.

Auswirkung auf den Datenbestand

- Wenn Sie einen Realm bei einer Satzart hinzufügen, die mit LOCATION MODE IS CALC definiert ist, so richtet BALTER - außer bei verteilbaren Listen - in diesem Realm einen Hashbereich für die Satzart ein.
- Wenn die Satzart Membersatzart einer verteilbaren Liste ist und diese mit LOCATION MODE IS CALC definiert ist, dann wird in einem Realm ein indirekter CALC-Bereich genutzt, der explizit durch die DETACHED WITHIN-Klausel der MODE IS LIST-Anweisung in der SSL oder ersatzweise durch den ersten Realmnamen der obigen WITHIN-Klausel bestimmt ist. Eine entsprechende Änderung bewirkt ein Neuanlegen des CALC-Bereiches.
  Wenn Sie ausschließlich einen oder mehrere Realms für eine Membersatzart einer verteilbaren Liste hinzufügen, bleibt die vorhandene Liste unverändert.
- Wenn Sie *realmname-1* ändern, so verlagert BALTER die DBTT der Satzart in den neu angegebenen Realm, falls dem nicht eine SSL-Angabe entgegen steht.

> Wenn Sie einen Realm in der WITHIN-Klausel weglassen, so dürfen in diesem Realm keine Sätze der betroffenen Satzart gespeichert sein. Sonst bricht BALTER - außer bei verteilbaren Listen - die Umstrukturierung ab, selbst wenn Sie einen anderen Realm für diese Sätze zur Verfügung stellen!
>
> Wenn Sie bei verteilbaren Listen einen Realm weglassen, führt dies zu einem Neuaufbau der Liste, auch wenn dieser Realm keine Sätze enthält.
>
> Wenn Sie eine verteilbare Liste entfernen, etwa durch Änderung der MODE-Klausel in POINTER-ARRAY, bleiben die Sätze der Membersatzart in dem Realm, in dem sie in der verteilbaren Liste gespeichert waren. Sofern Sie auf diese Sätze mit LOCATION MODE IS CALC zugreifen, müssen Sie dafür sorgen, daß nach Wegfall der Liste die AREA-ID jeweils korrekt versorgt ist. Um dabei entstehende Zugriffsprobleme zu vermeiden, können Sie vor der Um strukturierung mit der Online-Utility alle Seiten der Liste in einen Realm verlagern und mit dem Wegfall der Liste auch nur noch einen Realm für die Satzart deklarieren.
>
> Um Laufzeitverbesserungen des BALTER bei großen Datenbanken zu erreichen, können Sie folgendermaßen vorgehen:
>
> Wenn Sie die Realmzuordung einer Ownersatzart von einem auf zwei oder mehrere Realms erweitern, so sollten Sie in einem ersten Umstrukturierungsschritt die DETACHED-Angaben (auch die Standardwerte) auf DETACHED WITHIN *realmname* abändern für die Tabellen, die von der Ownersatzart abhängig sind. *realmname* bezeichnet den Realm, in dem die Ownersätze zurzeit gespeichert sind. Im zweiten Umstrukturierungsschritt nehmen Sie die auf einen Realm eingrenzenden DETACHED-Angaben wieder zurück.

bezeichner-2

dürfen Sie jederzeit ändern

Auswirkung auf den Datenbestand

Eine Änderung wirkt sich auf den Datenbestand nicht aus; lediglich in den Anwenderprogrammen müssen Sie sie berücksichtigen.

```
[SEARCH KEY IS feldname-3,...
   USING { CALC[ hashroutine] | INDEX} [ NAME IS name]
   DUPLICATES ARE[ NOT] ALLOWED] ... .
```

Bei der SEARCH-KEY Klausel sind alle Änderungen zugelassen; es ist erlaubt:

- bestehende SEARCH-Keys zu ändern
- neue SEARCH-Keys zu definieren
- nicht mehr benötigte SEARCH-Keys wegzulassen

feldname-3,...

Welches Datenfeld Sie als SEARCH-Key verwenden bzw. aus welchen Datenfeldern Sie den SEARCH-Key zusammensetzen, können Sie ohne Einschränkungen neu festlegen.

USING ...

alle Änderungen sind erlaubt

Auswirkung auf den Datenbestand

- CALC -> INDEX:
  BALTER baut eine mehrstufige SEARCH-Key-Tabelle auf und gibt den Speicherplatz des indirekten Hashbereichs frei.
- INDEX -> CALC:
  BALTER legt die SEARCH-Key-Tabelle als einen indirekten Hashbereich an und gibt den ursprünglich belegten Speicherplatz frei.
- Andere Hashroutine:
  BALTER richtet einen neuen Hashbereich ein und gibt den Speicherplatz des ursprünglichen Hashbereichs frei.

name diese Angabe können Sie weglassen, hinzufügen oder ändern

DUPLICATES...

für Duplikate gilt hier dasselbe wie für die LOCATION MODE-Klausel (siehe Abschnitt „Satz-Eintrag")

> ¡ BALTER prüft nur dann auf unzulässig doppelt vorhandene Schlüsselwerte und protokolliert diese, wenn er eine mehrstufige SEARCH-Key-Tabelle oder einen indirekten Hashbereich aufbauen muss.

```
{[stufennummer] satzelementname
   {PICTURE IS {maskenzeichenkette |
               LX (ganzzahl-1) DEPENDING ON feldname-4}

    TYPE IS {FIXED REAL {BINARY {15 | 31 | 63} |
                         DECIMAL[ ganzzahl-2[,ganzzahl-3]]} |
             CHARACTER[ ganzzahl-4[ DEPENDING ON feldname-5]] |
             DATABASE-KEY |
             DATABASE-KEY-LONG }

   [OCCURS ganzzahl-5 TIMES].}...
```

Den Aufbau von Satzarten dürfen Sie beliebig modifizieren. Beachten Sie dabei Folgendes:

- Länge der Satzart
  Satzarten, die in einer einstufigen Liste gespeichert sind, dürfen Sie nicht verlängern! Eine einstufige Liste bedeutet ORDER IS LAST, FIRST, PRIOR oder IMMATERIAL und MODE IS LIST.
- Anzahl und Reihenfolge der Felder
  Die Reihenfolge von Feldern dürfen Sie ändern, ebenso dürfen Sie Felder löschen.
  Neu definierte Felder initialisiert BALTER abhängig von Feldtyp:
  - alphanumerische Felder mit Leerzeichen
  - nationale Felder mit nationalen Leerzeichen (Unicode)
  - numerische Felder mit dem Wert Null

  Mit der FILL-Anweisung können Sie benutzerdefinierte Werte für die Initialisierung festlegen.
- Länge der Felder
  Die Felder können Sie verlängern oder verkürzen; abhängig vom Feldtyp geht BALTER folgendermaßen vor:
  - alphanumerische Felder:

    Beim Verlängern der Felder füllt BALTER rechts mit Leerzeichen auf, beim Verkürzen der Felder schneidet BALTER rechts entsprechend viele Zeichen ab.
  - nationale Felder:

    Beim Verlängern der Felder füllt BALTER rechts mit Leerzeichen (Unicode) auf, beim Verkürzen der Felder schneidet BALTER rechts entsprechend viele Zeichen ab.
  - numerische Felder:

    Unter Beachtung der Stellung des Dezimalpunkts füllt BALTER beim Verlängern der Felder links mit dem Wert Null auf und schneidet beim Verkürzen der Felder links entsprechend viele Zeichen ab, wobei relevante Ziffern verloren gehen können. Ein evtl. vorhandenes Vorzeichen bleibt erhalten, sofern die neue Feldbeschreibung ein Vorzeichen erlaubt.

- Database-Key-Felder
  Bei Database-Key-Feldern können Sie den Typ ändern von DATABASE-KEY in DATABASE-KEY-LONG und umgekehrt.

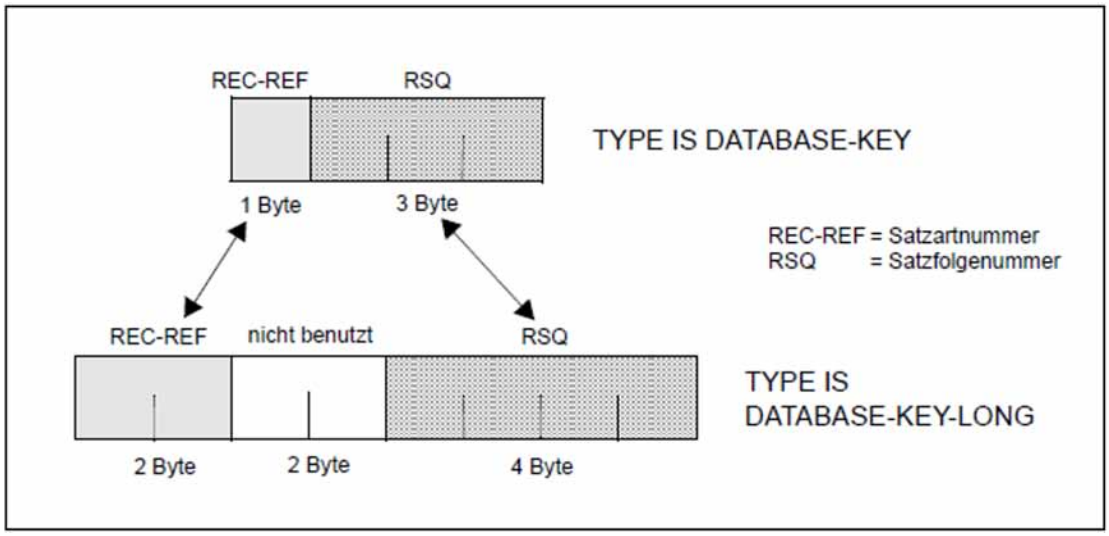

Bild 25: Typ von Database-Key-Feldern ändern

TYPE IS DATABASE-KEY -> TYPE IS DATABASE-KEY-LONG:
BALTER überträgt die 1 Byte lange Satzartnummer (REC-REF) des DATABASE-KEY-Feldes rechtsbündig in den entsprechenden 2 Byte langen Bereich des DATABASE-KEY-LONG-Feldes. Die 3 Byte lange Satzfolgenummer (RSQ) des DATABASE-KEY-Feldes wird rechtsbündig in den entsprechenden 4 Byte langen Bereich des DATABASE-KEY-LONG-Feldes übertragen.

TYPE IS DATABASE-KEY-LONG -> TYPE IS DATABASE-KEY:
BALTER überträgt das rechte Byte der 2 Byte langen Satzartnummer (REC-REF) des DATABASE-KEY-LONG-Feldes rechtsbündig in den entsprechenden 1 Byte langen Bereich des DATABASE-KEY-Feldes. Die rechten 3 Bytes der 4 Byte langen Satzfolgenummer (RSQ) des DATABASE-KEY-LONG-Feldes werden rechtsbündig in den entsprechenden 3 Byte langen Bereich des DATABASE-KEY-Feldes übertragen.

Da bei der Änderung von TYPE IS DATABASE-KEY-LONG in TYPE IS DATABASE-KEY bei REC-REF und RSQ links jeweils Stellen abgeschnitten werden, kommt es zu Datenverlust, falls REC-REF > 254 und/oder RSQ > 2^{24}-1. Bei Datenverlust gibt BALTER eine Warnung aus, die den ursprünglichen DATABASE-KEY-LONG-Wert enthält; der BALTER-Lauf wird nicht abgebrochen. Die Datenbank ist weiterhin physisch konsistent (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BCHECK). Die logische Konsistenz, d.h die Konsistenz der Anwenderdaten, muss von Ihnen überprüft und sichergestellt werden.

- Typ der Felder
  Wenn Sie den Typ eines **numerischen** Feldes in einen anderen numerischen Typ ändern (z.B. TYPE IS DECIMAL -> TYPE IS BINARY), konvertiert BALTER die Daten. Wenn Sie den Typ eines **ungepackten numerischen** Feldes in alphanumerisch (fester Länge) ändern, verfährt BALTER wie nachfolgend beschrieben. Bei allen anderen Typänderungen belegt BALTER das Feld entsprechend dem neuen Typ mit Leerzeichen oder dem Wert Null. Dies gilt insbesondere auch für alle Typänderungen von oder nach national.

  ungepackt numerisch -> alphanumerisch (feste Länge):
  Sofern erforderlich, konvertiert BALTER die Daten. Dabei verfährt BALTER wie folgt:

  1. BALTER überträgt die numerische Ziffernfolge linksbündig in das alphanumerische Zielfeld; führende Nullen bleiben erhalten.
     Ein evtl. vorhandenes Symbol „V“ (Dezimalpunkt) in der Definition des Ausgangsfeldes wird von BALTER ignoriert, d.h. übertragen werden der ganzzahlige Anteil **und** die Dezimalstellen.
     In der Definition des Ausgangsfeldes vorhandene Symbole „P“ (implizite Multiplikation mit 10) werden von BALTER so weit wie möglich berücksichtigt.

     Je nach Größe des alphanumerischen Zielfeldes verfährt BALTER wie folgt:

     - Enthält das Zielfeld weniger Stellen als die zu übertragende Ziffernfolge (inklusive berücksichtigter Symbole „P“), so werden von der Ziffernfolge rechts die überzähligen Stellen abgeschnitten.
     - Enthält das Zielfeld mehr Stellen als die zu übertragende Ziffernfolge (inklusive berücksichtigter Symbole „P“), so wird die Ziffernfolge rechts mit X‘40‘ auf Zielfeldlänge aufgefüllt.
  2. In der sedezimalen Darstellung der gemäß 1) erhaltenen Ziffernfolge wandelt BALTER das vorletzte Halbbyte (Vorzeichen) um in sedezimal „F“. Dies gilt insbesondere auch dann, wenn die Definition des Ausgangsfeldes das Symbol „S“ (Vorzeichen) enthält.

Beispiele

	Ausgangsfeld (ungepacktes numerisches Feld)			**Zielfeld (alphanumerisches Feld)**	
	Felddefinition	**Feldinhalt (sedezimal)**		**Felddefinition**	**Feldinhalt (sedezimal)**
1)	PIC 9999	F8 F1 F2 C3	->	PIC XXXX	F8 F1 F2 F3
2)	PIC S999PP	F5 F2 D3	->	PIC XXXXX	F5 F2 F3 F0 F0
3)	PIC S99V99	F1 F4 F3 D5	->	PIC XXXX	F1 F4 F3 F5
4)	PIC 9999	F1 F2 F3 F4	->	PIC XXX	F1 F2 F3
5)	PIC 999P	F5 F2 E3	->	PIC XXXXXX	F5 F2 F3 F0 40 40
6)	PIC S999V99	F0 F2 F3 F4 D5	->	PIC XXXXXX	F0 F2 F3 F4 F5 40
7)	PIC SV999	F0 F2 B3	->	PIC XX	F0 F2
8)	PIC V999	F7 F2 A3	->	PIC XX	F7 F2
9)	PIC S999PP	F0 F2 B3	->	PIC X	F0
10)	PIC 999PP	F8 F2 F3	->	PIC X	F8

- variables Feld
  In Satzarten dürfen Sie nur dann ein variables Feld hinzufügen oder ändern, wenn keine Sätze dieser Satzart in der Datenbank gespeichert sind (siehe "Zusammenfassung der Einschränkungen"). Sie können aber in Sätzen mit einem variablen Feld Felder fester Länge hinzufügen oder weglassen, wenn Sätze gespeichert sind. Auch können Sie die Länge solcher Felder ändern.
  Außerdem dürfen Sie alle Änderungen im Schema vornehmen, die zu einer Änderung des Systemteiles (SCD) führen.
- Stellung des Dezimalpunktes
  Ändern Sie die Stellung des Dezimalpunktes oder des Skalenfaktors, so verschiebt BALTER die Ziffern innerhalb des Feldes so, dass der alte Wert erhalten bleibt. Rechts oder links herausgeschobene Ziffern gehen verloren; BALTER rundet nicht.
- Wiederholungsfaktor
  Beim Verringern des Wiederholungsfaktors von Vektoren oder von Wiederholungsgruppen schneidet BALTER die Felder am Ende des Vektors bzw. der Wiederholungsgruppe ab. Beim Erhöhen des Wiederholungsfaktors initialisiert BALTER neu hinzukommende Felder typgerecht mit dem Wert Null bzw. Leerzeichen.
  Ob BALTER bei einer Änderung des Wiederholungsfaktors den Inhalt der Felder übernimmt, initialisiert oder wegfallen lässt, können Sie leicht nachprüfen:
  Die Umwandlungsroutine von BALTER verwendet bei indizierten Feldern sowohl für die alte als auch für die neue Definition eine dreistufige Indizierung. Stellen Sie das alte und das neue Feld jeweils mit einer dreistufigen Indizierung dar, so können Sie beide gut vergleichen.

> **!** **VORSICHT!**
> Verringern des Wiederholungsfaktors von Vektoren oder Wiederholungsgruppen kann den Verlust von Daten bedeuten.

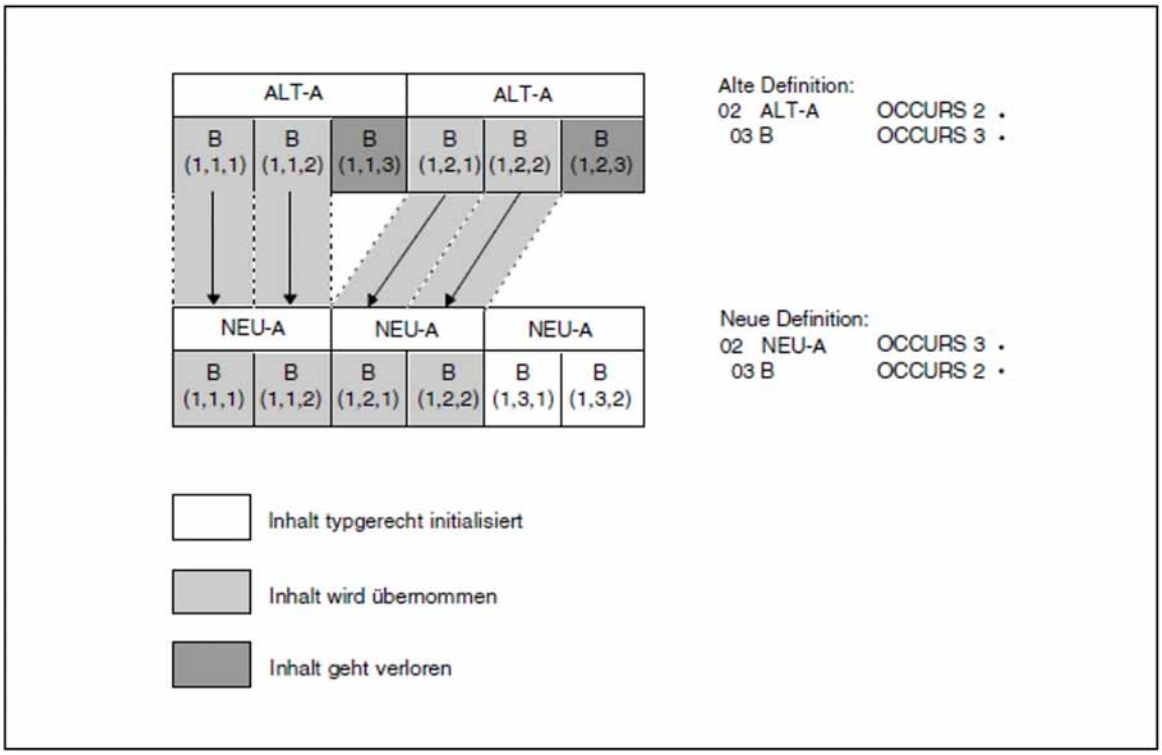

Bild 26: Veränderungen der Feldinhalte gespeicherter Sätze durch Modifizieren des Wiederholungsfaktors

6.1.4 Set-Eintrag

```
SET NAME IS setname
```

Beim Ändern des Set-Eintrags gilt Folgendes:

- Weglassen von Sets ist ohne Einschränkungen zugelassen.
- Hinzufügen von Sets ist erlaubt; außer beim Hinzufügen von SYSTEM-Sets bestehen jedoch Einschränkungen bei der Art der Set-Mitgliedschaft.
- Ändern von Sets ist nur mit Einschränkungen möglich.

```
[SET IS DYNAMIC]
```

Dynamische Sets dürfen Sie hinzufügen oder weglassen. Umwandeln eines Sets in einen dynamischen Set und umgekehrt ist nicht gestattet.

```
ORDER IS {LAST | FIRST | NEXT | PRIOR | IMMATERIAL |
          SORTED[ INDEXED[ NAME IS name]] BY
                 {DATABASE-KEY | DEFINED KEYS DUPLICATES ARE[ NOT] ALLOWED}}
```

Änderungen der ORDER-Klausel müssen Sie im Zusammenhang mit der MODE-Klausel der SSL betrachten (siehe Handbuch „Entwerfen und Definieren“, MODE-Klausel)!
Für Änderungen der ORDER-Klausel gilt:
Sind von der Membersatzart des zu ändernden Sets keine Sätze gespeichert, so sind alle Änderungen erlaubt; sind Sätze gespeichert, so dürfen Sie die ORDER-Klausel nicht uneingeschränkt ändern.

Welche Änderungen der ORDER-Klausel erlaubt sind, wenn Sätze der Membersatzart gespeichert sind, können Sie der folgenden Tabelle entnehmen:

altes Schema	**neues Schema**	ORDER IS {LAST | FIRST | NEXT | PRIOR | IMMATERIAL}	ORDER IS SORTED [INDEXED ...] .....
ORDER IS	MODE IS		
{LAST | FIRST | NEXT | PRIOR | IMMATERIAL}	POINTER-ARRAY	erlaubt ohne Einschränkung	erlaubt ohne Einschränkungen
	LIST	nur erlaubt, wenn die Membersatzart (einschließlich SCD) durch keine Änderung länger wird	
	CHAIN	nur erlaubt, wenn die SCD der Owner-/Membersatzart nicht vergrößert werden muss (siehe Tabelle 28)	
SORTED [INDEXED ...]	POINTER-ARRAY	nicht zugelassen	
	LIST	nicht zugelassen	
	CHAIN	nur erlaubt, wenn die SCD der Owner-/Membersatzart nicht vergrößert werden muss (siehe Tabelle 28)	

Tabelle 27: Änderungen der ORDER-Klausel, wenn Membersätze gespeichert sind

Wann und wie sich bei einem Set mit MODE IS CHAIN die Set Connection Data (SCD) ändert, zeigt die folgende Übersicht:

neues Schema **altes Schema**	ORDER IS	{FIRST | NEXT | PRIOR | IMMATERIAL| SORTED | [INDEXED...] ...}	{FIRST | NEXT | PRIOR | IMMATERIAL}	SORTED [INDEXED...] ...	LAST	
ORDER IS	MODE IS MODE IS	CHAIN	CHAIN LINKED TO PRIOR	CHAIN LINKED TO PRIOR	CHAIN	CHAIN LINKED TO PRIOR
{FIRST | NEXT | PRIOR | IMMATERIAL | SORTED | [INDEXED...]}	CHAIN	-	2) >Member-SCD >Owner-SCD	1) >Member-SCD >Owner-SCD	2) >Owner-SCD	2) >Owner-SCD >Member-SCD
	CHAIN LINKED TO PRIOR	1) <Owner-SCD <Member-SCD	-	-	1) <Member-SCD	-
LAST	CHAIN	1) <Owner-SCD	2) >Member-SCD	1) >Member-SCD	-	2) >Member-SCD
	CHAIN LINKED TO PRIOR	1) <Owner-SCD <Member-SCD	-	-	1) <Member-SCD	-

Tabelle 28: Änderungen der SCD bei einem Set mit MODE IS CHAIN

< wird kleiner

> wird größer

- keine Änderung der SCD-Länge

1) Änderung zugelassen, da SCD unverändert oder kürzer oder im neuen Schema ORDER IS SORTED [INDEXED...]...

2) Änderung nicht zugelassen, da SCD länger und im neuen Schema kein ORDER IS SORTED [INDEXED...]...

DUPLICATES ...

> ¡ hier gilt dasselbe wie für die LOCATION MODE-Klausel (siehe Abschnitt „Satz-Eintrag")

NAME IS *name*

diese Angabe können Sie beliebig ändern, hinzufügen oder weglassen

```
OWNER IS {satzname | SYSTEM}
```

Die OWNER-Klausel eines Sets zu verändern ist nicht gestattet!

```
MEMBER IS satzname {MANDATORY | OPTIONAL} {AUTOMATIC | MANUAL}
```

satzname

Eine neue Membersatzart anzugeben ist nicht erlaubt!

Set-Mitgliedschaft

Ändern Sie einen bestehenden Set und sind in der Datenbank

- keine Sätze der Membersatzart gespeichert, so dürfen Sie die Set-Mitgliedschaft beliebig modifizieren;
- Sätze der Membersatzart gespeichert, so dürfen Sie die Set-Mitgliedschaft nicht von OPTIONAL in MANDATORY AUTOMATIC umwandeln, wenn einige Membersätze keinem Owner zugeordnet sind.

> ¡ Wenn der Set sonst unverändert bleibt, führt diese Änderung zu einer inkonsistenten Datenbank; wenn der Set wegen sonstiger Änderungen verarbeitet werden muss, führt sie zum abnormalen Ende von BALTER.

Definieren Sie einen neuen Set und sind in der Datenbank

- keine Sätze der Membersatzart gespeichert, so dürfen Sie die Set-Mitgliedschaft beliebig wählen;
- Sätze der Membersatzart gespeichert, dann dürfen Sie - außer bei SYSTEM-Sets - als Set-Mitgliedschaft nicht MANDATORY AUTOMATIC definieren.BALTER kann Membersätze nicht automatisch Ownersätzen zuordnen, wenn der Set nicht singulär ist, d.h. Sie müssen selbst entscheiden, welchem Ownersatz welche Membersätze zuzuordnen sind und dies per Programm mit der CONNECT-Anweisung durchführen.

Auswirkung auf den Datenbestand

Eine Änderung der Set-Mitgliedschaft wirkt sich nicht auf den Datenbestand aus. Sie müssen allerdings in Ihren Programmen beim Speichern oder Löschen von Datensätzen eine eventuell geänderte Set-Mitgliedschaft berücksichtigen.

```
[{ASCENDING | DESCENDING} KEY IS feldname-1,...]
```

Alle Änderungen sind erlaubt.

Auswirkung auf den Datenbestand

BALTER baut die betroffenen Adresslisten, Listen oder Ketten entsprechend der geänderten Sortierkriterien neu auf.

Dabei schließt der Neuaufbau einer Liste auch ein Umspeichern der Membersätze mit ein, da BALTER diese in eine neue Sortierreihenfolge bringen muss.

```
[SEARCH KEY IS feldname-2,... USING {CALC | INDEX}[ NAME IS name]
         DUPLICATES ARE[ NOT] ALLOWED]...
```

Für SEARCH-Keys auf Setebene bestehen dieselben Modifikationsmöglichkeiten wie auf Satzartebene (siehe SEARCH KEY-Klausel ändern, "Satz-Eintrag").

```
[SET OCCURRRENCE SELECTION IS THRU
    {CURRENT OF SET |
     LOCATION MODE OF OWNER
              [ALIAS FOR {feldname-3 | bezeichner-1} IS bezeichner-2]...} ].
```

Ändern der SET OCCURRENCE SELECTION-Klausel ist zugelassen.

Auswirkung auf den Datenbestand

Eine Änderung wirkt sich nicht auf den Datenbestand aus; Sie müssen sie lediglich in Ihren DB-Anwendungen berücksichtigen.

6.2 SSL ändern

Das Laufzeitverhalten Ihrer DB-Anwendungen wird im Wesentlichen durch die Speicherstruktur in Ihrer Datenbank bestimmt. Wenn Sie Ihre Datenbank optimieren wollen, müssen Sie die Speicherstruktur an die zeitkritischen Anwendungen anpassen, indem Sie die SSL-Klauseln ändern. So können Sie die Laufzeiten Ihrer DB-Anwendung erheblich verbessern, auch bei gleich bleibendem Schema und gleich bleibenden Programmen.

Auch wenn Sie Ihre bisherige SSL verwenden, müssen Sie diese neu übersetzen, da BGSIA sonst die Standardwerte der SSL-Klauseln einsetzt!

Folgende Änderungen der Speicherstruktur sind zugelassen:

- auf Satzartebene
  - DBTTs in andere Realms verlagern
  - die PLACEMENT OPTIMIZATION neu festlegen
  - bei SEARCH-Key-Tabellen Lage, Typ und Seitenanzahl für die Reorganisation neu definieren
  - Datensätze komprimieren oder entkomprimieren (mit Einschränkungen)
- auf Setebene
  - die MODE-Klausel ändern (mit Einschränkungen)
  - die Adresslisten oder Listen eines Sets verlagern
  - die Owner eines Sets zusätzlich mit ihren Adresslisten oder Listen verketten oder eine bestehende Verkettung aufheben
  - den Umfang ändern, in dem die Tabellen der Set-Occurrences eines Sets angelegt bzw. erweitert werden
  - die Seitenanzahl für die Reorganisation von Tabellen eines Sets neu festlegen
  - bei SEARCH-Key- und Sort-Key-Tabellen Lage, Typ und Seitenanzahl für die Reorganisation neu definieren
  - Member mit ihrem zugehörigen Owner verketten oder eine bestehende Verkettung aufheben

Alle Klauseln der SSL sind wahlweise anzugeben, d.h. Sie dürfen sie nach Bedarf erstmalig angeben, ändern oder weglassen. Für jede Klausel, die Sie weglassen, setzt BGSIA den Standardwert ein.

Die Klauseln der Schema-DDL und der SSL unterliegen bei einer Umstrukturierung den gleichen Regeln wie beim Definieren der Datenbank (siehe Handbuch „Entwerfen und Definieren“). Haben Sie Klauseln der Schema-DDL geändert, müssen Sie alle damit zusammenhängenden Klauseln der SSL so anpassen, dass sie diesen Regeln entsprechen.

Die an der Speicherstruktur möglichen Modifikationen und ihre Auswirkungen auf den Datenbestand sind im Folgenden ausführlich beschrieben.

6.2.1 Schema-Eintrag

```
STORAGE STRUCTURE OF SCHEMA schemaname.
```

schemaname

muss mit dem Namen im Schema-Eintrag der neuen Schema-DDL übereinstimmen

6.2.2 Satz-Eintrag

```
RECORD NAME IS satzname
```

satzname

muss eine Satzart der neuen Schema-DDL bezeichnen

```
[DATABASE-KEY-TRANSLATION-TABLE[ IS ganzzahl-1][ WITHIN realmname-1]]
```

ganzzahl-1

Die Anzahl der DBTT-Einträge lässt sich z.B. mit dem Dienstprogramm BREORG ändern. BALTER ignoriert generell eine Änderung der Größenangabe.

realmname-1

dürfen Sie modifizieren

Auswirkung auf den Datenbestand

BALTER verlagert die DBTT in den angegebenen Realm.

```
[PLACEMENT OPTIMIZATION FOR SET setname]
```

Jede beliebige Änderung ist zugelassen.

Auswirkung auf den Datenbestand

Die Position bereits gespeicherter Datensätze ändert BALTER nicht; neu zu speichernde Datensätze legt der DBH entsprechend der von BALTER vermerkten neuen Angaben in der Datenbank ab.

Spezifizieren Sie diese Klausel erstmalig für eine in einem direkten Hashbereich gespeicherte Satzart, so legt BALTER einen indirekten Hashbereich an, ändert aber nicht die Lage der betroffenen Sätze, sondern wandelt lediglich die CALC-Seiten des ehemaligen Hashbereichs in normale Datenseiten um.

```
[POPULATION IS {ganzzahl-2 WITHIN realmname-2},...]
```

ganzzahl

Die Größe eines Hashbereichs lässt sich mit dem Dienstprogramm BREORG ändern; BALTER ignoriert eine Änderung der Größenangabe.

Ausnahme

BALTER muss auf Grund einer der folgenden Änderungen den Hashbereich neu anlegen:

- von direktem zu indirektem Hashbereich oder umgekehrt
- der Hashroutine
- der Zusammensetzung des CALC-Keys
- des Realms, in dem der Hashbereich liegen soll

BALTER verwendet *ganzzahl-2* zum Berechnen der neuen Größe des Hashbereichs.

```
[INDEX NAME IS name[ PLACING IS WITHIN realmname-3]
      [TYPE IS {DATABASE-KEY-LIST |
                REPEATED-KEY
                    [DYNAMIC REORGANIZATION SPANS ganzzahl-3 PAGES]}]]
```

PLACING IS ...

SEARCH-Key-Tabellen dürfen Sie in andere Realms verlagern.

TYPE IS ...

Den Aufbau der SEARCH-Key-Tabellen dürfen Sie modifizieren, indem Sie die TYPE-Klausel in DATABASE-KEY-LIST oder in REPEATED-KEY ändern.

DYNAMIC REORGANIZATION

Die Seitenanzahl für die Reorganisation können Sie beliebig neu festlegen.

Auswirkung auf den Datenbestand

Eine Änderung der Seitenanzahl wirkt sich nicht sofort bei der Umstrukturierung aus, sondern erst im Datenbankbetrieb beim Speichern weiterer Sätze.

```
[COMPRESSION FOR ALL ITEMS].
```

Bisher nicht komprimiert gespeicherte Satzarten in Zukunft komprimiert zu speichern und umgekehrt, ist mit Einschränkungen gestattet:

- Ändern von nicht komprimiert in komprimiert:

  Bereits gespeicherte Sätze komprimiert BALTER nicht - entfernt also keine leeren Datenfelder aus Datensätzen - sondern erweitert nur die Set-Connection-Data (SCD) der Sätze der betroffenen Satzart um ein vier Byte langes Kompressionsfeld.
  Da BALTER hierdurch jeden Satz der Satzart um vier Byte verlängert, muss er die Sätze, die in den bisher von dieser Satzart belegten Seiten keinen Platz mehr haben, in andere Seiten umspeichern.
- Ändern von komprimiert in nicht komprimiert:

  Die Änderung hängt davon ab, wie die bereits gespeicherten Sätze der betroffenen Satzart eingetragen sind:
  - Sind die Sätze alle in voller Länge eingetragen (wie z.B. durch BINILOAD), so löscht BALTER das Kompressionsfeld in der SCD der Datensätze. Da die Sätze hierdurch um vier Byte kürzer werden, schiebt BALTER sie in der jeweiligen Seite zusammen, sodass innerhalb der Seiten Platz frei wird
  - Sind die Sätze komprimiert, also nicht in voller Länge eingetragen, so bricht BALTER die Umstrukturierung ab!

    Komprimiert gespeicherte Datensätze kann BALTER generell nicht verarbeiten, auch wenn die Änderung nicht das Komprimieren betrifft:

    Die Sätze selbst dürfen nicht verändert werden, weder der Benutzerteil noch die Systeminformation. Tabellen oder Hashbereiche, die als Schlüssel Felder dieser Satzart enthalten, kann BALTER nicht aufbauen.

6.2.3 Set-Eintrag

```
SET NAME IS setname
```

setname

muss mit dem Namen eines Sets der neuen Schema-DDL übereinstimmen

```
[MODE IS
  {CHAIN[ LINKED TO PRIOR] |
  {POINTER-ARRAY | LIST} {ATTACHED TO OWNER |
                         DETACHED[ WITHIN realmname-1]
                                 [ WITH PHYSICAL LINK]}}]
```

Die Mode-Klausel müssen Sie im Zusammenhang mit der ORDER-Klausel der Schema-DDL betrachten (siehe ORDER-Klausel ändern, "Set-Eintrag")!

Haben Sie in der ORDER-Klausel der neuen Schema-DDL

- SORTED[INDEXED] festgelegt, so dürfen Sie die Verkettungsmethode in der MODE-Klausel beliebig ändern;
- nicht SORTED[INDEXED] festgelegt, so sind Änderungen der Verkettungsmethode verboten, wenn bereits Sätze der Membersatzart gespeichert sind.

Wenn Sie eine verteilbare Liste (MODE IS LIST) in eine Adressliste (MODE IS POINTER-ARRAY) ändern, verbleiben die Membersätze in dem Realm, in dem sie aktuell in der Liste gespeichert waren. Gegebenenfalls müssen Sie die Zugriffslogik Ihrer Anwendungsprogramme anpassen.

Bei einer Änderung in MODE IS LIST wird eine Liste aufgebaut. Sätze der betroffenen Satzart werden möglicherweise in einen anderen Realm verlagert.

ATTACHED/DETACHED

Eine Änderung von ATTACHED in DETACHED und umgekehrt ist erlaubt, mit der Einschränkung, dass Sätze, die in einer Liste gespeichert sind, nicht in einen anderen Realm verlagert werden dürfen.

Auswirkung auf den Datenbestand

- Bereits existierende Tabellen speichert BALTER nur dann um, wenn er sie auf Grund einer Änderung der Angabe WITHIN *realmname-1* in einen anderen Realm verlagern muss.
- Die Lage einer Liste wird ohne DETACHED WITHIN-Klausel durch die Lage des Owners bestimmt. Insofern kann es bei Änderung von Pointer-Array nach Liste zu einer Verlagerung der Sätze kommen, und bisherige Programme mit Direktzugriffen (FIND4) müssen angepasst werden.

WITHIN *realmname-1*

Eine Adressliste dürfen Sie in einen anderen Realm verlagern. Eine Liste dürfen Sie - außer bei verteilbaren Listen - nur dann verlagern, wenn keine Membersätze gespeichert sind.

Bei verteilbaren Listen bestimmt der *realmname-1* implizit die Lage des Tabellenteils (Stufe-1- bis Stufe-N-Seiten) der Liste und die Lage eines indirekten CALC-Bereiches bei LOCATION MODE IS CALC, wenn Sie nicht explizit in der SSL bei der MODE IS LIST-Anweisung die Lage festlegen.

Sie können die Tabellenseiten in einen anderen Realm verlagern, auch wenn in der verteilbaren Liste Membersätze gespeichert sind.

Auswirkungen auf den Datenbestand:

Die Liste und ein eventueller CALC-Bereich werden neu aufgebaut. Die Membersätze werden dabei über die beteiligten Realms annähernd gleich verteilt.

WITH PHYSICAL LINK

Eine Adressliste oder Liste, die getrennt vom Owner gespeichert ist, dürfen Sie erstmalig zusätzlich mit dem zugehörigen Owner verketten oder eine bestehende Verkettung aufheben.

Auswirkung auf den Datenbestand

- Legen Sie einen zusätzlichen Adressverweis an, so erweitert BALTER die Set Connection Data (SCD) der Ownersätze um die physische Adresse der höchsten Stufe der Tabelle. Da BALTER hierdurch die Ownersätze verlängert, speichert es die Sätze, für die in den bisher belegten Seiten kein Platz mehr ist, in der Datenbank um.
- Löschen Sie den zusätzlichen Adressverweis, so entfernt BALTER in den SCD der Ownersätze die physische Adresse der höchsten Stufe der Tabelle.

```
[POPULATION IS ganzzahl-1[ INCREASE IS ganzzahl-2]]
```

ganzzahl-1

dürfen Sie beliebig verändern. BALTER berücksichtigt diese Angabe in Verbindung mit einer entsprechenden Anweisung FILLING WITH POPULATION. Eine Änderung wirkt sich auch beim Speichern neuer Ownersätze bzw. beim Löschen von Membersätzen aus.
Bestehende Tabellen verändert BALTER nicht.

ganzzahl-2

dürfen Sie beliebig verändern. Eine Änderung wirkt sich erst aus, wenn beim Speichern neuer Membersätze Tabellenerweiterungen notwendig werden.

```
[DYNAMIC REORGANIZATION SPANS ganzzahl-3 PAGES]
```

Die Seitenanzahl für das Reorganisieren der Tabellen des Sets dürfen Sie beliebig ändern.

```
[INDEX NAME IS name
   [ PLACING IS {ATTACHED TO OWNER | DETACHED [WITHIN realmname-2]}
    [TYPE IS {DATABASE-KEY-LIST |
              REPEATED-KEY [DYNAMIC REORGANIZATION SPANS ganzzahl PAGES]}]]
```

Für die INDEX-Klausel des Set-Eintrags gilt sinngemäß dasselbe wie für die INDEX-Klausel des Satz-Eintrags (siehe INDEX-Klausel ändern, "Satz-Eintrag").

ATTACHED/DETACHED

eine Änderung von ATTACHED in DETACHED oder umgekehrt ist ohne Einschränkungen erlaubt

Auswirkung auf den Datenbestand

Eine Änderung wirkt sich genauso aus wie bei der MODE-Klausel
(siehe MODE-Klausel ändern, "Set-Eintrag").

WITHIN *realmname-2*

dürfen Sie hinzufügen oder weglassen oder *realmname-2* ändern

Auswirkung auf den Datenbestand

Bei einer Änderung verlagert BALTER die Tabellen in den angegebenen Realm.

```
[MEMBER IS PHYSICALLY LINKED TO OWNER].
```

Eine zusätzliche Verkettung der Membersätze mit dem zugehörigen Ownersatz dürfen Sie erstmalig verlangen oder aufheben.

Auswirkung auf den Datenbestand

- Geben Sie die Klausel erstmalig an, so erweitert BALTER die SCD der Membersätze um den physischen Adressverweis auf den zugehörigen Ownersatz. Da BALTER die Membersätze hierbei verlängert, muss er die Sätze, für die in den bisher belegten Seiten kein Platz mehr ist, in der Datenbank umspeichern.
- Entfernen Sie die Klausel aus Ihrer SSL, so verkürzt BALTER die SCD der Membersätze um den physischen Adressverweis auf den zugehörigen Ownersatz.

6.3 Zusammenfassung der Einschränkungen

Dieses Kapitel bietet eine Übersicht, welche Änderungen in der Schema-DDL und der SSL nicht bzw. nur mit Einschränkungen zugelassen sind, wenn Sätze einer betroffenen Satzart gespeichert sind. Für diese Änderungen gibt BALTER im Analyseprotokoll Warnungen aus!

- Komprimierung
  Komprimiert gespeicherte Sätze kann BALTER nicht verarbeiten. BALTER bricht die Umstrukturierung ab, wenn Sie die Satzart vor dem Umstrukturieren nicht entladen haben und eine der folgenden Änderungen durchführen:
  - Ändern des Benutzer- oder Systemteils der Satzart
  - Ändern des Aufbaus von Tabellen oder Hashbereichen, die Felder dieser Satzart als Schlüssel enthalten
- Variable Felder
  Falls zu einer Satzart mit variablem Feld Sätze in der Datenbank gespeichert sind, gibt es Einschränkungen bei der Umstrukturierung. BALTER bricht die Umstrukturierung in folgenden Fällen ab:
  - Hinzufügen eines Feldes variabler Länge
  - Entfernen eines Feldes variabler Länge
  - Änderung der Maximallänge eines Feldes variabler Länge.
- Listen
  Satzarten, die in einer einstufigen Liste gespeichert sind, dürfen Sie nicht verlängern. Eine einstufige Liste bedeutet ORDER IS LAST, FIRST, NEXT, PRIOR oder IMMATERIAL und MODE IS LIST.

  Nicht verteilbare Listen dürfen Sie nicht in andere Realms verlagern.

  Verteilbare Listen können Sie beliebig verlagern. Wenn Sie ausschließlich Realms hinzufügen, bleibt die Liste unverändert und der Umstrukturierungsprozess ist insgesamt sehr schnell durchführbar.
- zyklische Setstrukturen
  Soll BALTER bei allen Sets einer zyklischen Setstruktur (siehe Handbuch „Entwerfen und Definieren“, Zyklus) eine Änderung durchführen, bei der die Ownersatzart vor der Membersatzart verarbeitet werden muss, so führt dies zu einer Deadlock-Situation. Solche Änderungen sind Aufbauen einer neuen Kette oder die Membersätze zusätzlich mit dem jeweiligen Ownersatz verketten bei Sets, bei denen die Ownersatzart auf Grund weiterer Änderungen verlagert werden muss.

  Planen Sie derartige Änderungen bei einer zyklischen Setstruktur, so müssen Sie die Umstrukturierung auf zwei Schritte aufteilen.

6.3.1 Schema-DDL-Änderungen

Schema-Eintrag

keine Einschränkungen

Realm-Eintrag

Einen Temporären Realm in einen nicht Temporären Realm umzuwandeln, bzw. umgekehrt, ist nicht zugelassen.

Satz-Eintrag

WITHIN-Klausel

Lassen Sie einen Realm in der WITHIN-Klausel weg, so dürfen - außer bei verteilbaren Listen - in diesem Realm keine Sätze der betroffenen Satzart gespeichert sein.

Satzlänge

Satzarten, die Member in einer einstufigen Liste sind, dürfen Sie nicht verlängern.

Set-Eintrag

DYNAMIC-Klausel

Umwandeln eines Sets in einen dynamischen Set und umgekehrt, ist nicht gestattet.

ORDER-Klausel

Sind von der Membersatzart des zu ändernden Sets Sätze gespeichert, so gilt Folgendes:

- bei ORDER IS SORTED[INDEXED] und MODE IS POINTER-ARRAY/LIST dürfen Sie die ORDER-Klausel generell nicht in ORDER IS LAST, FIRST, NEXT, PRIOR oder IMMATERIAL ändern;
- bei beliebiger ORDER-Klausel und MODE IS CHAIN dürfen Sie die ORDER-Klausel nur dann in ORDER IS LAST, FIRST, NEXT, PRIOR oder IMMATERIAL ändern, wenn dadurch weder die SCD der Ownersatzart noch die der Membersatzart länger wird;
- bei ORDER IS LAST, FIRST, NEXT, PRIOR oder IMMATERIAL und MODE IS LIST dürfen Sie die ORDER-Klausel nur dann in ORDER IS LAST, FIRST, NEXT, PRIOR oder IMMATERIAL ändern, wenn dadurch die Membersatzart einschließlich SCD nicht länger wird.

OWNER-Klausel

Die OWNER-Klausel eines Sets dürfen Sie nicht ändern.

MEMBER-Klausel

satzname: eine neue Membersatzart anzugeben ist nicht erlaubt.

Set-Mitgliedschaft:

- ändern Sie einen bestehenden Set und sind in der Datenbank Sätze der Membersatzart gespeichert, so dürfen Sie die Set-Mitgliedschaft nicht von OPTIONAL in MANDATORY AUTOMATIC umwandeln, wenn einige gespeicherte Membersätze keinem Owner zugeordnet sind.
- definieren Sie einen neuen Set und sind in der Datenbank Sätze der Membersatzart gespeichert, so dürfen Sie – außer bei SYSTEM-Sets – als Set-Mitgliedschaft nicht AUTOMATIC definieren.

6.3.2 SSL-Änderungen

Schema-Eintrag

keine Einschränkungen

Satz-Eintrag

DBTT-Klausel

Die Anzahl der DBTT-Einträge lässt sich z. B. mit dem Dienstprogramm BREORG ändern. BALTER ignoriert generell eine Änderung der Größenangabe.

POPULATION-Klausel

Die Größe eines Hashbereichs lässt sich mit dem Dienstprogramm BREORG ändern. BALTER ignoriert eine Änderung der Größenangabe, es sei denn, es muss den Hashbereich auf Grund weiterer Änderungen neu anlegen.

COMPRESSION-Klausel

Komprimiert gespeicherte Sätze kann BALTER nicht verarbeiten, also weder entkomprimieren noch umspeichern, sortieren oder löschen.

Set-Eintrag

MODE-Klausel

Lautet die ORDER-Klausel im neuen Schema LAST, FIRST, NEXT, PRIOR oder IMMATERIAL so dürfen Sie die Verkettungsmethode in der MODE-Klausel nur dann ändern, wenn von der Membersatzart keine Sätze gespeichert sind.

6.4 Konsistenz der Datenbank prüfen

Eine konsistente Datenbank ist die wichtigste Voraussetzung für eine erfolgreiche Umstrukturierung. Mit dem Dienstprogramm BCHECK sollten sie daher prüfen, ob Ihre Datenbank konsistent ist (siehe Handbuch „Sichern, Informieren und Reorganisieren“).

> ¡ Stellt BCHECK Inkonsistenzen fest, so sollten Sie auf keinen Fall Ihre Datenbank umstrukturieren, bevor Sie sie nicht repariert haben; sie verschlimmern sonst unter Umständen den Fehler und erschweren so ein Reparieren der inkonsistenten Datenbank!

6.5 Freien Speicherplatz prüfen

Zum Anpassen des Datenbestands an das geänderte Schema bzw. die geänderte Speicherstruktur beansprucht BALTER zusätzlichen freien Speicherplatz

- zum Anlegen neuer Tabellen, Hashbereiche oder DBTTs
- zum Umspeichern von Datensätzen, Tabellen, Hashbereichen oder DBTTs.

Reicht der freie Speicherplatz in den Realms dafür nicht aus, dann erweitert BALTER die betroffenen Realms automatisch, sofern dies möglich ist (Voraussetzung: Sekundärzuweisung für Speicherplatz > 0). Näheres hierzu siehe Handbuch „Datenbankbetrieb“, Automatische Realm-Erweiterung durch Dienstprogramme).

Falls keine automatische Realm-Erweiterung möglich ist (Sekundärzuweisung für Speicherplatz = 0 oder kein Plattenspeicher mehr verfügbar), bricht BALTER die Umstrukturierung ab! Ihre Datenbank ist dadurch inkonsistent und Sie müssen sie auf den Stand vor Beginn der Umstrukturierung zurücksetzen und wieder von vorne beginnen.

Realms ohne automatische Realm-Erweiterung

Wenn ein oder mehrere Realms mit einer Sekundärzuweisung = 0 eingerichtet sind, z.B. weil Sie keine automatische Realm-Erweiterung wünschen, müssen Sie vor dem Umstrukturieren sicherstellen, dass der verfügbare Speicherplatz ausreicht. Dazu müssen Sie sich von BSTATUS ausgeben lassen, wie viel Speicherplatz in den verschiedenen Realms Ihrer Datenbank noch frei ist und abschätzen, ob der freie Platz für die geplante Umstrukturierung ausreicht.

> ¡ Wichtig ist dabei, dass Sie dies möglichst frühzeitig tun, denn BREORG kann die Realms Ihrer Datenbank nur erweitern, solange BCHANGE die Datenbank noch nicht für die Umstrukturierung vorbereitet hat.
> Sie können zu diesem Zweck auch das Analyseprotokoll zu Hilfe nehmen (siehe Abschnitt „Beschreibung des Analyseprotokolls (REPORT-Phase)").

BALTER wählt die Verarbeitungsreihenfolge so, dass er als Erstes die Elemente aus der Datenbank löscht, die im neuen Schema nicht mehr vorhanden sind. Dieser Platz ist somit bei späteren Aktionen (Aufbauen neuer Tabellen, Anlegen neuer Hashbereiche, Umspeichern von Sätzen) wieder nutzbar. Ebenso kann der Platz, der beim Umspeichern einer Satzart frei wird, nach Abschluss des Umspeicherns wieder verwendet werden.
Durch ungünstige Konstellationen kann es jedoch vorkommen, dass zuerst der Platz belegt wird, der hinten im Realm frei ist, und vorne im Realm Lücken bleiben. In so einem Fall können Sie durch Reorganisieren mit dem Dienstprogramm BREORG nach dem Umstrukturieren Hashbereiche und Tabellen in die freien vorderen Seiten verlagern:

- Hashbereiche mit REORGANIZE-CALC
- Tabellen mit REORGANIZE-SET

Anschließend können Sie die Realms um die am Ende frei gewordenen Seiten verkleinern.

Die Übersichten auf den folgenden Seiten sollen Ihnen helfen, die Größe des Speicherplatzes abzuschätzen, der zum Umstrukturieren in den Realms Ihrer Datenbank unbedingt noch frei sein muss. Stellen Sie dabei fest, dass ein Realm zu wenig Speicherplatz frei hat, so müssen Sie ihn mit BREORG erweitern, bevor Sie BCHANGE zum Vorbereiten der Umstrukturierung starten.

DBTT

Den freien Speicherplatz, den BALTER zum Neuanlegen oder Umspeichern einer DBTT benötigt, listet er im Analyseprotokoll auf.

Sie können die neue Größe einer DBTT aber auch anhand der in der Tabelle angegebenen Formeln berechnen.

Änderung in der Datenbank	**U r s a c h e**	**S p e i c h e r b e d a r f**		
		Formel (siehe "Berechnungs-formeln")	**SSL-Klausel, die die Größe bestimmt**	**Standard-größe**
neue DBTT anlegen	neue Satzart definiert	1	DBTT-Klausel	1 Seite
DBTT umspeichern	Anzahl der DBTT-Spalten verändert (nur beim Ownersatz), durch • geänderte Verkettungs-Methode in der MODE-Klausel • hinzugekommenen/weggelassenen SEARCH-Key (Setebene) • ändern der Anzahl der Sets, in denen diese Satzart Owner ist DBTT verlagert durch: • neuen <realmname-1> in der WITHIN-Klausel der DDL (Satzartebene) • neuen *realmname-1* in der DBTT-Klausel der SSL (Satzartebene)	2	-	ursprüngliche Anzahl der DBTT-Einträge

Tabelle 29: Speicherbedarf bei DBTT-Änderungen

Hashbereiche

Die Anzahl der Seiten, die BALTER zum Einrichten eines Hashbereichs benötigt, listet er im Analyseprotokoll auf.

Sie können die Größe der Hashbereiche aber auch anhand von Formeln berechnen! Ein Hashbereich belegt immer zusammenhängende Seiten eines Realm.

Änderung in der Datenbank	Ursache	Speicherbedarf		
		Formel, siehe "Berechnungsformeln"	SSL-Klausel, die die Größe bestimmt	Standardgröße
direkten Hashbereich bzw. indirekten Hashbereich (Primärschlüssel) einrichten	• neue Satzart definiert mit LOCATION CALC • LOCATION MODE in CALC geändert • bei LOCATION CALC die Hashroutine oder den CALC-Key geändert • Satzart mit LOCATION CALC verlängert/verkürzt	3 bzw. 4	POPULATION-Klausel	1 Seite
indirekten Hashbereich (Sekundärschlüssel) einrichten	• neuen SEARCH-Key mit USING CALC definiert • Definition eines SEARCH-Key in USING CALC geändert • CALC-Key oder Hashroutine eines SEARCH-Key verändert • indirekten Hashbereich in einen anderen Realm verlagert	4	DBTT-Klausel	1 Seite
direkten Hashbereich umwandeln in indirekten Hashbereich (Primärschlüssel)	Eine mit LOCATION CALC definierte Satzart • ist Owner oder Member in einem Set, für den erstmalig PLACEMENT OPTIMIZATION an gegeben wird • wird Member in einer Liste • wird im neuen Schema erstmalig komprimiert gespeicher	4	POPULATION-Klausel	1 Seite
indirekten Hashbereich umwandeln in direkten Hashbereich (Primärschlüssel)	Bei einer mit LOCATION CALC definierten Satzart • wird PLACEMENT OPTIMIZATION weggelassen bei allen Sets, in denen diese Satzart Member ist • wird komprimiertes Speichern gestrichen • wird die MODE-Klausel eines Set, in dem diese Satzart Member ist, geändert von MODE IS LIST in POINTER-ARRAY/CHAIN	3		

Tabelle 30: Speicherbedarf bei Änderungen, die Hashbereiche betreffen

Tabellen

Verlagern von Tabellen

Verlagert BALTER Tabellen - nur einstufige Tabellen kann BALTER als ganzes verlagern - in einen anderen Realm, so entspricht der dort benötigte Platz dem ursprünglich von der Tabelle belegten Platz.

Neuaufbau von Tabellen:

- SYSTEM-Set oder TYPE IS DATABASE-KEY-LIST:
  Jede Set-Occurrence-Tabelle belegt mindestens eine Seite.

  Standard-Set und nicht TYPE IS DATABASE-KEY-LIST:
  Jede Set-Occurrence-Tabelle belegt nur so viel Platz, wie sie benötigt.

  BALTER speichert Tabellen desselben Sets so Platz sparend ab, dass mehrere Tabellen in derselben Seite liegen können.

Tabelle	**Änderung in der Schema-DDL bzw. der SSL**	**Änderung in DBB**		**S p e i c h e r b e d a r f**
		Tabelle (n) neu aufbauen	**Tabelle (n) verlagern**	
indizierte SEARCH-Key-Tabelle (Satzartebene)	neuen SEARCH-Key mit USING INDEX definiert	X	-	min. 1 Seite max. siehe Formel 5
	Schlüsselfeld geändert	X	-	
	Typ der Tabelle geändert REPEATED-KEY <--> DATABASE-KEY-LIST	X	-	
	Definition eines SEARCH-Key in USING INDEX umgewandelt	X	-	
	SEARCH-Key-Tabelle in einen anderen Realm verlagert	X	-	

indizierte SEARCH-Key-Tabelle (Setebene)	Typ der Tabelle geändert REPEATED-KEY --> DATABASE-KEY-LIST oder Schlüsselfeld geändert beim Typ DATABASE-KEY-LIST	X	-	min. 1 Seite pro Set-Occurrence-Tabelle bei SYSTEM-Set: min. 1 Seite max. siehe Formel 5
	neuen SEARCH-Key mit USING INDEX definiert	X	-	bei SYSTEM-Set: min. 1 Seite max. siehe Formel 5 bei Standard-Set: Mindestangabe nicht möglich; BALTER speichert mehrere Set-Occurrence-Tabellen zusammen, wenn sie klein genug sind,und in denselben Realm kommen sollen.
	Schlüsselfeld geändert	X	-	
	Typ der Tabelle geändert DATABASE-KEY-LIST --> REPEATED-KEY	X	-	
	Definition eines SEARCH-Key in USING INDEX umgewandelt	X	-	
	SEARCH-Key-Tabelle in einen anderen Realm verlagert	X	-	
indizierte Adressliste	MODE-Klausel in POINTER-ARRAY geändert	X	-	max. Größe für eine einzelne Set-Occurrence-Tabelle siehe Formel 5
	ASC/DSC-Key geändert	X	-	
	Adressliste in einen anderen Realm verlagert	X	-	
	neuen Set mit MODE IS POINTER-ARRAY definiert	X	-	
indizierte Sort-Key-Tabelle (MODE IS CHAIN)	neuen Set definiert mit MODE IS CHAIN und ORDER SORTED INDEXED	X	-	
	Definition eines Sets geändert in MODE IS CHAIN und ORDER SORTED INDEXED	X	-	
	ASC-/DESC-Key geändert	X	-	
	Tabelle in einen anderen Realm verlagert	X	-	

nicht indizierte Adressliste	Adressliste in einen anderen Realm verlagert	-	X	ursprüngliche Größe
indizierte Liste	MODE-Klausel in LIST geändert	X	-	bei SYSTEM-Set: min. 1 Seite max. siehe Formel 5 bei Standard-Set: Mindestangabe nicht möglich; BALTER speichert mehrere Set-Occurrence-Tabellen zusammen, wenn sie klein genug sind, und in enselben Realm kommen sollen. max. Größe für eine einzelne Set-Occurrence-Tabelle siehe Formel 5
	ASC-/DESC-Key geändert	X	-	
	neuen Set mit MODE IS LIST definiert	X	-	

Tabelle 32: Speicherbedarf bei Tabellenänderungen

> ¡ Wird in einem leeren SYSTEM-Set eine Adressliste, Liste oder Kette SORTED INDEXED neu definiert, so legt BALTER eine leere Tabelle in der Größe einer Seite dafür an.
>
> Folgende Änderungen sind nur dann zulässig, wenn keine Membersätze vorliegen:
>
> - Neuaufbau einer einstufigen Liste oder einer einstufigen Adressliste (bewirkt z.B. durch MODE-Definition, Änderung ORDER-Klausel)
> - Verlagern einer einstufigen Liste in demselben Realm (z.B. durch Satzverlängerung)
> - Verlagern einer Liste in einen anderen Realm

Umspeichern von Sätzen

BALTER unterscheidet beim Umspeichern von Sätzen zwischen teilweiser und vollständiger Verlagerung:

- Vollständige Verlagerung

  liegt vor, wenn das neue Schema eine besondere Ablageform zwingend vorschreibt. Dies ist der Fall bei LOCATION MODE IS CALC bei direkten Hashbereichen und bei Satzarten, die in einer Liste gespeichert sind.

  Vollständige Verlagerung führt BALTER in folgenden Fällen durch:

  - Sie verwenden im neuen Schema eine eigene Hashroutine statt der Standard-Hashroutine
  - Sie ändern im neuen Schema bei einem direkten Hashbereich den CALC-Key
  - Sie lassen im neuen Schema den Grund weg für eine indirekte Hash-Speicherung
  - Sie führen im neuen Schema LOCATION MODE IS CALC neu ein
  - Sie ändern im neuen Schema den ASC-/DESC-Key einer Liste
  - Sie definieren im neuen Schema MODE IS LIST neu

Bei vollständiger Verlagerung speichert BALTER alle Sätze in neue Seiten um und gibt die alten Seiten frei. Das heißt jedoch, dass während des BALTER-Laufs das Doppelte des für die Speicherung der Sätze notwendigen Platzes vorhanden sein muss.

Verlängern Sie eine Satzart mit besonderer Ablageform, so führt dies zu vollständiger Verlagerung, auch wenn sich an der Ablageform selbst nichts ändert.

- Teilweise Verlagerung

  liegt vor, wenn sich die Sätze verlängern (Benutzer- oder Systemteil) und aus diesem Grund mehr Seiten zur Speicherung benötigen.

  Bei teilweiser Verlagerung bleiben so viele Sätze in den bisherigen Seiten wie möglich, den Rest speichert BALTER in neue Seiten um.

- Verkürzen von Sätzen

  Verkürzen Sie eine Satzart mit besonderer Ablageform, so führt dies wie beim Verlängern einer solchen Satzart zu vollständiger Verlagerung, auch wenn sich an der Ablageform selbst nichts ändert.

  Verkürzen Sie eine Satzart ohne besondere Ablageform, so schiebt BALTER jeweils die Sätze innerhalb einer Seite zusammen. Es wird also Platz innerhalb der einzelnen Seiten frei, BALTER speichert die Sätze aber nicht in andere Seiten um, sodass keine ganzen Seiten frei werden.

BALTER druckt den Platz, den er zum Umspeichern von Sätzen benötigt, nicht im Analyseprotokoll aus. Mit Hilfe von Tabelle 33 und der anschließend dargestellten Berechnungsformeln können Sie jedoch den benötigten Speicherplatz zum Teil berechnen.

Änderung	**U r s a c h e**		**S p e i c h e r b e d a r f**		
			Formel (siehe "Berechnungs-formeln")	**SSL-Klausel, die die Größe bestimmt**	**Standard-größe**
vollständige Verlagerung	direkter Hash-bereich	LOCATION MODE in CALC geändert	3	POPULATION-Klausel	1 Seite
		Hashroutine/CALC-Key geändert			
		indirekten in direkten Hashbereich umgewandelt (siehe Tabelle 30)			
	indizierte Liste	neuen Set mit MODE IS LIST definiert	bei SYSTEM-Set: min. 1 Seite max. siehe Formel 5 bei Standard-Set: Mindestangabe nicht möglich; BALTER speichert mehrere Set-Occurrence-Tabellen zusammen, wenn sie klein genug sind, und in denselben Realm kommen sollen. max. Größe für eine einzelne Set-Occurrence-Tabelle siehe Formel 5		
		MODE-Klausel in LIST geändert			
		ASC-/DESC-Key geändert			

<table>
<tr><td rowspan="8">teilweise
Verlagerung
(Satzart verlängert)</td><td colspan="2">Benutzerteil verlängert</td><td rowspan="8">BALTER trägt so viele Sätze wie möglich in den bisherigen Seiten ein, den Rest verlagert er in andere Seiten.

Da die Sätze in den meisten Fällen gestreut gespeichert sind, ist eine Berechnung des erforderlichen Speicherplatzes nicht möglich
(außer bei Satzarten mit besonderer Ablageform).</td></tr>
<tr><td rowspan="7">Systemteil verlängert</td><td>Satzart komprimiert</td></tr>
<tr><td>Owner verkettet mit seiner Tabelle</td></tr>
<tr><td>Member verkettet mit seinem Owner</td></tr>
<tr><td>MODE IS CHAIN eingeführt
(siehe Tabelle 28, "Set-Eintrag")</td></tr>
<tr><td>Owner-/Membersatzart: neuen Set mit MODE IS CHAIN hinzugefügt</td></tr>
<tr><td>bei MODE IS CHAIN hinzugefügt:
LINKED TO PRIOR
oder
ORDER IS LAST
(siehe Tabelle 28, "Set-Eintrag")</td></tr>
<tr><td>Membersatzart: neuen Set hinzugefügt</td></tr>
</table>

<table>
<tr><td rowspan="7">(keine
Verlagerung)

Satzart
verkürzt</td><td colspan="2">Benutzerteil verkürzt</td><td rowspan="7">BALTER schiebt jeweils dieSätze innerhalb einer Seite zusammen
(außer bei einer Satzart mit besonderer Ablageform).

Zusätzlicher freier Speicherplatz ist - außer bei einer Satzart mit besonderer Ablageform - nicht erforderlich.</td></tr>
<tr><td rowspan="6">Systemteil
verkürzt</td><td>Satzart entkomprimiert</td></tr>
<tr><td>die Verkettung
des Owner mit seiner
Tabelle aufgehoben</td></tr>
<tr><td>die Verkettung
des Members mit
seinem
Owner aufgehoben</td></tr>
<tr><td>mit MODE IS CHAIN
definierte Sets gelöscht</td></tr>
<tr><td>bei MODE IS CHAIN
weggelassen: LINKED
TO PRIOR und
ORDER IS LAST
(siehe Tabelle 28, "Set-Eintrag")</td></tr>
<tr><td>die MODE-Klausel
geändert
von MODE IS CHAIN
in POINTER-ARRAY /
LIST
(siehe Tabelle 28, "Set-Eintrag)</td></tr>
<tr><td>Set gelöscht, in dem
die Satzart Member war</td></tr>
</table>

Tabelle 33: Speicherbedarf beim Umspeichern von Sätzen

6.5.1 Berechnungsformeln

Mit den nachfolgend angegebenen Berechnungsformeln können Sie folgende Werte berechnen:

- DBTT-Größe für neu angelegte DBTT
- DBTT-Größe für umgespeicherte DBTT
- Größe eines direkten Hashbereichs
- Größe eines indirekten Hashbereichs
- Größe einer mehrstufigen SEARCH-Key-Tabelle auf Satzartebene

1. DBTT-Größe berechnen für neu angelegte DBTT:

   `2044 / LENGTH = einträge-pro-seite` [1] `(bei 2048 Byte Seitenlänge)`

   `3980 / LENGTH = einträge-pro-seite` [1] `(bei 4000 Byte Seitenlänge)`

   `8076 / LENGTH = einträge-pro-seite` [1] `(bei 8096 Byte Seitenlänge)`

   `ganzzahl / einträge-pro-seite = seitenanzahl` [2]

   [1] Ergebnis ist abzurunden auf eine ganze Zahl

   [2] Ergebnis ist aufzurunden auf eine ganze Zahl

   seitenanzahl

   Anzahl der DBTT-Seiten

   einträge-pro-seite

   Anzahl der DBTT-Einträge pro Seite

   ganzzahl

   Anzahl der geplanten Sätze aus DBTT-Klausel

   LENGTH

   Länge eines DBTT-Eintrags; diesen Wert enthält das BGSIA-Protokoll unter der Überschrift „DBTT-INFORMATION“
   (siehe Handbuch „Sichern, Informieren und Reorganisieren")

2. DBTT-Größe berechnen für umgespeicherte DBTT:

   2044 / *LENGTH-neu* = *einträge-pro-seite-neu* [1] (bei 2048 Byte Seitenlänge)

   3980 / *LENGTH-neu* = *einträge-pro-seite-neu* [1] (bei 4000 Byte Seitenlänge)

   8076 / *LENGTH-neu* = *einträge-pro-seite-neu* [1] (bei 8096 Byte Seitenlänge)

   alte-gesamtzahl-einträge / einträge-pro-seite-neu = *seitenanzahl* [2]

   [1] Ergebnis ist abzurunden auf eine ganze Zahl

   [2] Ergebnis ist aufzurunden auf eine ganze Zahl

   seitenanzahl

   Anzahl der DBTT-Seiten

   alte-gesamtzahl-einträge

   Anzahl der Einträge in der ursprünglichen DBTT; diesen Wert können Sie mit BSTATUS ermitteln

   einträge-pro-seite-neu

   Anzahl der DBTT-Einträge pro Seite in der neuen DBTT

   LENGTH-neu

   Länge eines Eintrags in der neuen DBTT

   LENGTH-neu = 4 x (*n* + 1)

   n

   Anzahl aller Tabellen der Sets, in denen die Satzart Ownersatzart ist

3. Größe berechnen für direkten Hashbereich:

   Die Anzahl der Überlaufseiten, die BALTER mit anlegen muss, ist hierbei nicht berücksichtigt!

   `(`*`seitenlänge`* `- 30) / (`*`satzlänge`* `+` *`calc-key-länge`* `+ 15) =` *`einträge-pro-seite`* [1)]

   `(ganzzahl` `- 1) /` *`einträge-pro-seite`* `+ 1 =` *`seitenanzahl`* [2)]

   [1)] Ergebnis ist abzurunden auf eine ganze Zahl

   [2)] Ergebnis ist aufzurunden auf die nächsthöhere Primzahl, wenn sich keine Primzahl ergibt

   seitenlänge

   Seitenlänge in der Datenbank (2048/4000/8096 Byte)

   seitenanzahl

   Anzahl der Seiten des Hashbereichs

   calc-key-länge

   Länge des CALC-Key;
   diesen Wert können Sie dem BGSIA-Protokoll entnehmen unter der Überschrift 'CALC-INFORMATION' in der Spalte LENGTH
   (siehe Handbuch „Sichern, Informieren und Reorganisieren")

   ganzzahl

   Anzahl der zu speichernden Sätze laut POPULATION-Klausel (Satzartebene)

   satzlänge

   Länge der Satzart inkl. SCD;
   diesen Wert können Sie dem BGSIA-Protokoll entnehmen unter der Überschrift „RECORD-INFORMATION " in der Spalte LENGTH
   (siehe Handbuch „Sichern, Informieren und Reorganisieren")

   einträge-pro-seite

   Anzahl der Einträge (Sätze oder CALC-Index-Einträge) pro Seite

4. Größe berechnen für indirekten Hashbereich:

   Die Anzahl der Überlaufseiten, die BALTER mit anlegen muss, ist hierbei nicht berücksichtigt.

   `(seitenlänge` `- 30 ) /` *`(calc-key-länge`* `+ 7) =` *`einträge-pro-seite`* [1)]

   `(ganzzahl` `- 1) /` *`einträge-pro-seite`*`) =` *`seitenanzahl`* [2)]

   [1)] Ergebnis ist abzurunden auf eine ganze Zahl

   [2)] Ergebnis ist aufzurunden auf die nächsthöhere Primzahl, wenn sich keine Primzahl ergibt

   ganzzahl

   Anzahl der zu speichernden Sätze

   - laut POPULATION-Klausel (Primärschlüssel)
   - laut DBTT-Klausel (Sekundärschlüssel).

   Übrige Bedeutungen siehe Punkt 3.

5. Größe berechnen für mehrstufige SEARCH-Key-Tabelle (Satzartebene):

   Diese Formel geht von geringfügig vereinfachten Annahmen aus und liefert deshalb eine Abschätzung und nicht den exakten Zahlenwert.

   Bei 2048 Byte Seitenlänge:

   - falls ein Füllgrad angegeben wurde:

     ```
            anzahl-search-keys                  2002 - a
     ------------------------------------ * ---------- = seitenanzahl
     max(1, (2002 / a * füllgrad / 100)) 1)    2002 - 2a
     ```

     1) Ergebnis des ersten Bruchs und Ergebnis der Multiplikation sind jeweils abzurunden auf eine ganze Zahl

   - falls kein Füllgrad angegeben wurde:

     ```
     anzahl-search-keys    2002 - a
     ----------------- * ---------- = seitenanzahl
     (2002 / a ) -1) 2)    2002 - 2a
     ```

     2) Ergebnis des Bruchs ist abzurunden auf eine ganze Zahl

   wobei: a = *search-key-l* + 7

   Bei 4000 Byte Seitenlänge:

   - falls ein Füllgrad angegeben wurde:

     ```
            anzahl-search-keys                  3950 - b
     ------------------------------------ * ---------- = seitenanzahl
     max(1, (3950 / b * füllgrad / 100)) 1)    3950 - 2b
     ```

     1) Ergebnis des ersten Bruchs und Ergebnis der Multiplikation sind jeweils abzurunden auf eine ganze Zahl

   - falls kein Füllgrad angegeben wurde:

     ```
     anzahl-search-keys    3950 - b
     ----------------- * ---------- = seitenanzahl
     (3950 / b ) -1) 2)   3950 - 2b
     ```

     2) Ergebnis des Bruchs ist abzurunden auf eine ganze Zahl

   wobei: b = *search-key-l* + 10

Bei 8096 Byte Seitenlänge:

- falls ein Füllgrad angegeben wurde:

$$\frac{\textit{anzahl-search-keys}}{\max(1, (8046 / b * \textit{füllgrad} / 100))\ ^{1)}} * \frac{8046 - b}{8046 - 2b} = \textit{seitenanzahl}$$

[1] Ergebnis des ersten Bruchs und Ergebnis der Multiplikation sind jeweils abzurunden auf eine ganze Zahl

- falls kein Füllgrad angegeben wurde:

$$\frac{\textit{anzahl-search-keys}}{(8046 / b) -1\ ^{2)}} * \frac{8046 - b}{8046 - 2b} = \textit{seitenanzahl}$$

[2] Ergebnis des Bruchs ist abzurunden auf eine ganze Zahl

wobei: b= *search-key-l* + 10

seitenanzahl

Anzahl der benötigten Seiten für die gesamte Set-Occurrence-Tabelle

anzahl-search-keys

Anzahl der Schlüssel in der Tabelle

füllgrad

Füllgrad in Prozent
(siehe FILLING-KLausel, "FILLING (Füllgrad von Tabellenseiten festlegen)")

search-key-l(änge)

Länge des SEARCH-Key;
diesen Wert können Sie dem BGSIA-Protokoll entnehmen unter der Überschrift „KEY-INFORMATION (NO CALC SEARCH KEY'S)“ in der Spalte LENGTH
(siehe Handbuch „Sichern, Informieren und Reorganisieren“).

6.6 Sicherungsmaßnahmen und Verhalten im Fehlerfall

Eine Umstrukturierung verändert neben der Benutzerdatenbank auch die Compilerdatenbank und die HASHLIB.

Treten während der Umstrukturierung Fehler auf, sodass Sie auf die Datenbank mit dem Stand vor Beginn der Umstrukturierung zurückgreifen müssen, müssen Sie daher auch auf die Realms der Compilerdatenbank und die HASHLIB zurückgreifen.

6.6.1 Datenbank sichern

Vor Beginn des Umstrukturierungsprozesses, also vor dem Starten von BCHANGE, müssen Sie das After-Image-Logging ausschalten (BALTER schreibt keine After-Images) und Ihre Datenbank komplett oder im benötigten Umfang sichern.

Zu sichern sind in jedem Fall

dbname.HASHLIB
dbname.COSSD
dbname.DBDIR
dbname.DBCOM

Bezüglich der Benutzerrealms gibt es zwei Möglichkeiten:

- Sie sichern alle Benutzerrealms vor Beginn des Umstrukturierungsprozesses, also
  dbname.realmname-1
  .
  .
  dbname.realmname-n
- Sie ermitteln in einem Analyselauf mit den Anweisungen
  REPORT IS YES
  EXECUTION IS NO,
  welche Benutzerrealms benötigt werden, und sichern nur diese zusätzlich vor der Ausführungsphase des BALTER.

  Ermittlung der benötigten Benutzerrealms:

  Nach der Sicherung von *dbname*.HASHLIB, *dbname*.COSSD, *dbname*.DBDIR und *dbname*.DBCOM führen Sie den Umstrukturierungsprozess einschliesslich des Analyselaufes des BALTER mit REPORT IS YES und EXECUTION IS NO durch. Dem Analyseprotokoll des BALTER können Sie unter REPORT OF ADDED, DELETED AND NEEDED AREAS entnehmen, welche Benutzerrealms benötigt werden.
  Evtl. werden auch Realms benötigt, weil Ankersätze von singulären Sets neu aufgebaut werden müssen.
  BALTER versucht dabei unnötige Verlagerungen zwischen den Realms zu vermeiden, um damit auch den notwendigen Sicherungsaufwand bei den Benutzerrealms zu beschränken.

Sicherung der benötigten Benutzerrealms:

Vor der Durchführung der Umstrukturierung mit BALTER und der Anweisung EXECUTION IS YES sichern Sie dann die in der Analysephase ermittelten Benutzerrealms
dbname.realmname-i
.
.
dbname.realmname-j

Weitere Informationen zur Sicherung einer Datenbank finden Sie im Handbuch „Datenbankbetrieb“, Datenbank sichern und wiederherstellen im Fehlerfall.

6.6.2 Datenbank rekonstruieren

Bricht während des Umstrukturierungsprozesses ein Programm die Verarbeitung mit „ABNORMAL END“ ab, so müssen Sie, je nach Schwere des aufgetretenen Fehlers eine der folgenden Maßnahmen durchführen:

- die Ausführung des abgebrochenen Programms wiederholen
- auf die angelegte Sicherung zurückgreifen und den Umstrukturierungsprozess wiederholen

Wann es erforderlich ist, auf eine Sicherung der Datenbank zurückzugreifen und den Umstrukturierungsprozess zu wiederholen und wann es genügt, das abgebrochene Programm zu wiederholen, ist bei den einzelnen Programmen beschrieben!

Folgende Tabelle zeigt, welche Programme im Laufe der Umstrukturierung Dateien oder Realms der Datenbank verändern:

	HASHLIB	**DBDIR**	**DBCOM**	**DBCOM.O**	**COSSD**	**COSSD.O**	**Benutzerrealms, auf die zugegriffen werden muss**
BCHANGE	-	LS	LS	S	L	S	-
DDL-Compiler	-	LS	LS	-	S	-	-
SSL-Compiler	-	LS	LS	-	S	-	-
BGSIA	-	LS	LS	-	-	-	-
LMS	S	-	-	-	-	-	-
BALTER (Analysephase)	-	L	L	L	-	-	-
BALTER (Umstrukturierungsphase)	L	LS	L	L	-	-	LS
DDL-Compiler (Subschemata)	-	LS	LS	-	S	L	-
BGSSIA	-	LS	L	-	-	-	-

Tabelle 34: Zugriff auf die Dateien und Realms der Datenbank während einer Umstrukturierung

L lesender Zugriff

S schreibender Zugriff

- kein Zugriff

Für die Rekonstruktion der Datenbank stehen Ihnen folgende Möglichkeiten zur Verfügung:

- Sie können die Schattendatenbank in eine Originaldatenbank umwandeln durch Umbenennen mit dem MODIFY-FILE-ATTRIBUTES-Kommando.
- Sie können die ARCHIVE-Sicherung einspielen und evtl. den Datenbanknamen ändern mit dem MODIFY-FILE-ATTRIBUTES-Kommando. Wenn die ARCHIVE-Sicherung online erstellt wurde, müssen Sie sie evtl. mit dem Dienstprogramm BMEND nachfahren (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BMEND).

Weitere Informationen zur Rekonstruktion einer Datenbank finden Sie im Handbuch „Datenbankbetrieb“, Datenbank sichern und wiederherstellen im Fehlerfall.

Maßnahmen bei Speicherbedarf

Beendet sich eines der Programme mit Datenbank-Status (DBSTATUS) 14802 bzw. 14804, so müssen Sie

- mit BREORG den betroffenen Realm bzw. die betroffene Satzart erweitern,
- das Programm neu starten und ggf. die nur teilweise erzeugte Information unter Nutzung der DELETE-Anweisung von BGSSIA und evtl. auch der DELETE-Anweisung des DDL-Compilers löschen.

6.7 Compilerdatenbank vorbereiten mit BCHANGE

Die Aufgabe von BCHANGE beim Umstrukturieren einer Datenbank ist vergleichbar mit der Aufgabe von BCREATE beim Einrichten einer Datenbank: BCHANGE bereitet die Compilerdatenbank auf die Aufnahme des neuen Schemas vor. Im Einzelnen führt BCHANGE folgende Vorarbeiten zum Umstrukturieren aus:

- Sicherstellen der alten SIA im DBDIR und das DBDIR auf die Aufnahme einer neuen SIA vorbereiten, sodass im DBDIR nach dem BGSIA-Lauf für das neue Schema eine neue und eine alte SIA gespeichert sind. BALTER benötigt beide SIAs beim Anpassen des Datenbestands an das neue Schema, um die Abweichungen des neuen Schemas vom alten Schema erkennen zu können.
  Achten Sie daher darauf, dass vor dem BCHANGE-Lauf genügend freie Seiten im DBDIR zur Verfügung stehen bzw. durch Sekundärzuweisung > 0 eine automatische Realmerweiterung möglich ist.
- Löschen aller Benutzer-SSIAs im DBDIR.
- Sicherstellen des alten DBCOM in der Datei *dbname*.DBCOM.O und Neuformatieren des DBCOM.

  BALTER benötigt die Schema-Informationen des alten und des neuen DBCOM zum Untersuchen der geplanten Änderungen.
- Sicherstellen des alten COSSD in der Datei *dbname*.COSSD.O.

  Der DDL-Compiler benötigt nach der Umstrukturierung den alten COSSD für das Übernehmen der Subschemata. Sie sollten daher die Datei *dbname*.COSSD.O erst löschen, nachdem Sie alle weiterhin benötigten Subschemata übersetzt bzw. übernommen haben.

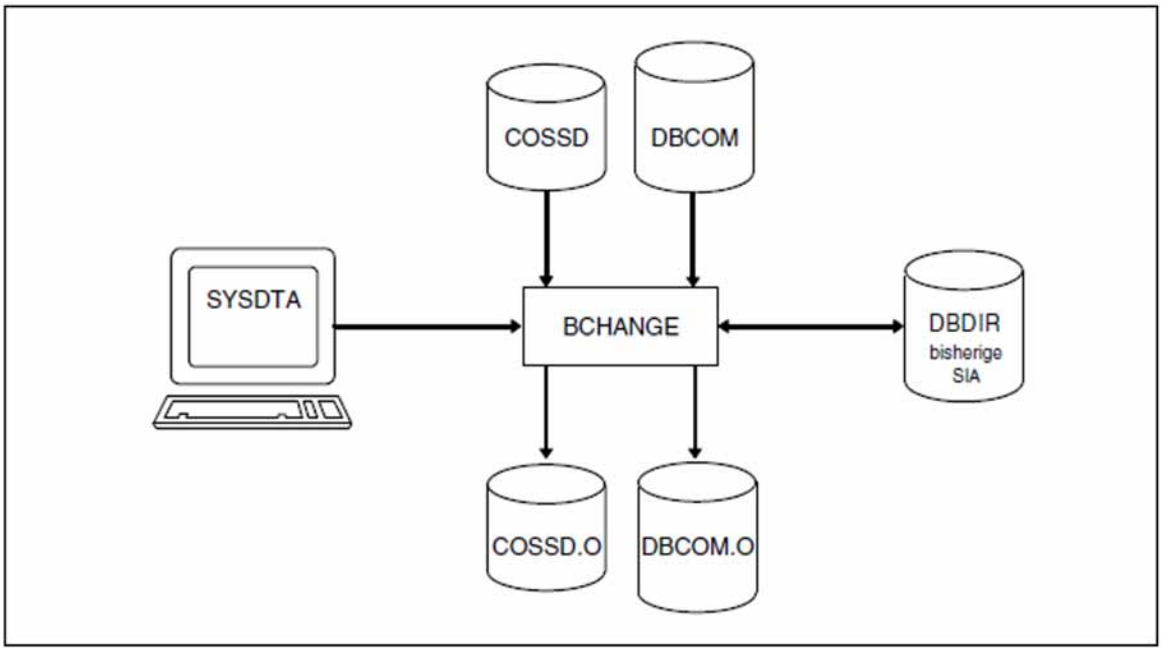

Bild 27: Systemumgebung beim Vorbereiten der Compilerdatenbank

BCHANGE legt die Kopien des DBCOM und des COSSD automatisch auf gemeinschaftlicher Platte an. Ein CREATE-FILE-Kommando vor dem Starten von BCHANGE zum Einrichten der beiden Dateien ist nur erforderlich, wenn die Kopien auf privater Platte angelegt werden sollen.
Je nach Größe der Dateien ist es allerdings ratsam, sie - auch wenn sie auf gemeinschaftlicher Platte liegen sollen - mit einem CREATE-FILE-Kommando mit SPACE-Angabe einzurichten (siehe auch „Maximalgröße für UDS/SQL-Dateien“, "Dateien und Realms einer UDS/SQL-Datenbank").

BCHANGE erweitert bei Bedarf automatisch die Realms der bearbeiteten Datenbank. Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme).

BCHANGE berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

Kommandofolge zum Starten von BCHANGE

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

Das Dienstprogramm BCHANGE starten Sie in der Kennung, unter der die Datenbank katalogisiert ist, mit folgenden Kommandos:

```
01 [/CREATE-FILE FILE-NAME=dbname.DBCOM.0 ...]
02 [/CREATE-FILE FILE-NAME=dbname.COSSD.0 ...]
03 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
04 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
05 /START-UDS-BCHANGE
```

01,02 Siehe Abschnitt „Compilerdatenbank einrichten".

04 Die angegebene Version von BCHANGE wird ausgewählt.
Die Angabe der Version wird generell empfohlen, da mehrere UDS/SQL-Versionen parallel installiert sein können.

05 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen BCHANGE gestartet werden.

¡ BCHANGE hat keine Anweisungen!

6.8 Schema-DDL übersetzen

Wenn Sie mit dem Dienstprogramm BCHANGE die Compilerdatenbank für die Aufnahme eines neuen Schemas vorbereitet haben, so müssen Sie als Nächstes mit dem DDL-Compiler Ihre aktuelle Schema-DDL übersetzen, auch wenn Sie die Schema-DDL nicht verändert haben.

Der Übersetzungsvorgang entspricht dem beim Einrichten der Datenbank.

Nach dem Übersetzen der Schema-DDL existieren:

- ein alter und ein neuer DBCOM
- eine alte SIA im DBDIR
- ein alter und ein neuer COSSD

Kommandofolge zum Übersetzen der aktuellen Schema-DDL

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

Die aufgeführten Kommandos sind im Abschnitt „Schema-DDL übersetzen" ausführlich beschrieben!

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-DDL
04  SOURCE IS 'schemadatei'
05  END
06 /ASSIGN-SYSDTA TO=*SYSCMD
```

> ¡ Achten Sie darauf, dass der DDL-Compiler die Übersetzung mit „NORMAL END“ abschließt.
>
> Wenn er sie mit „ABNORMAL END“ beendet, so müssen Sie die Übersetzung mit korrigierten DDL-Klauseln wiederholen.

6.9 SSL übersetzen

Wahlweise können Sie, nachdem Sie Ihre Schema-DDL übersetzt haben, mit dem SSL-Compiler eine neue SSL übersetzen.

Führen Sie keine SSL-Übersetzung durch, so werden die Standardwerte für die Speicherstruktur eingesetzt. Wollen Sie Ihre bisher definierte Speicherstruktur beibehalten, so müssen Sie Ihre ursprünglichen SSL-Klauseln neu übersetzen!

Der Übersetzungsvorgang entspricht dem beim Einrichten der Datenbank (siehe Abschnitt „SSL übersetzen").

Kommandofolge zum Übersetzen der SSL

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme". Dort finden Sie auch die möglichen Aliasnamen für den Aufruf).

Die aufgeführten Kommandos sind im Abschnitt „SSL übersetzen" ausführlich beschrieben!

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /START-UDS-SSL
04 SOURCE IS 'ssl-datei'
05 END
06 /ASSIGN-SYSDTA TO=*SYSCMD
```

> ¡ Achten Sie darauf, dass der SSL-Compiler die Übersetzung mit „NORMAL END“ abschließt. Wenn er die Übersetzung mit „ABNORMAL END“ beendet, so müssen Sie folgende Maßnahmen durchführen:
>
> - bei Fehlern in den SSL-Klauseln:
>   die fehlerhaften SSL-Klauseln korrigieren und die SSL-Übersetzung wiederholen;
> - bei Fehlern in den DDL-Klauseln:
>   - die fehlerhaften DDL-Klauseln korrigieren
>   - das fehlerhafte Schema löschen in einem DDL-Lauf mit der Anweisung DELETE SCHEMA *schemaname*
>   - den Umstrukturierungsprozess ab „Übersetzen der Schema-DDL“ wiederholen.

6.10 Neue SIA erzeugen und in das DBDIR eintragen mit BGSIA

Nach dem erfolgreichen Übersetzen der Schema-DDL und (wahlweise) der SSL müssen Sie mit dem Dienstprogramm BGSIA die SIA des neuen Schemas erzeugen und in das DBDIR eintragen.

Die gesicherte SIA des alten Schemas bleibt im DBDIR stehen, sodass das DBDIR nach dem BGSIA-Lauf die SIA des alten und des neuen Schemas enthält. BALTER benötigt beide zum Anpassen des Datenbestands an das geänderte Schema.

Der BGSIA-Lauf entspricht dem beim Einrichten der Datenbank (siehe „Schema Information Area (SIA) erzeugen mit BGSIA"). Das Modul UDSHASH, das BGSIA erzeugt, müssen Sie nach dem BGSIA-Lauf in die HASHLIB eintragen.

Wenn Sie mit eigenen Hashroutinen arbeiten, so müssen Sie diese spätestens vor dem Starten von BALTER mit EXECUTION IS YES zusätzlich mit den Attributen RMODE=ANY und AMODE=ANY in die HASHLIB eintragen.

SIA erzeugen und in das DBDIR eintragen

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /DELETE-SYSTEM-FILE FILE-NAME=*OMF
04 /START-UDS-BGSIA
05 GENERATE SCHEMA schemaname 06 [DISPLAY[ SCHEMA schemaname]]
07 END
```

Das Modul UDSHASH in die HASHLIB eintragen

```
01 /START-LMS
02 //OPEN-LIB LIB=dbname.HASHLIB,MODE=*UPDATE
03 //ADD-ELEMENT FROM-FILE=*OMF,TO-ELEMENT=*LIBRARY-ELEMENT(TYPE=R)
04 //END
```

6.11 Schemaänderungen analysieren und Datenbestand anpassen mit BALTER

Die Änderungen am Datenbankschema zu analysieren und den Datenbestand an das geänderte Datenbankschema anzupassen, ist Aufgabe des Dienstprogramms BALTER. BALTER steuert diese Vorgänge in zwei Phasen:

- in der Analysephase analysiert BALTER die Änderungen am Datenbankschema
- In der optionalen REPORT-Phase gibt BALTER das Analyseprotokoll aus
- in der Umstrukturierungsphase passt BALTER den Datenbestand und auch die Definition der Datenbank an das geänderte Schema an.

BALTER können Sie nur dann erfolgreich ablaufen lassen, wenn Sie zuvor mit dem Dienstprogramm BGSIA die neue SIA erzeugt und ins DBDIR eingetragen haben (siehe Abschnitt „Neue SIA erzeugen und in das DBDIR eintragen mit BGSIA"). Andernfalls beendet sich der BALTER-Lauf mit der Meldung „BGSIA HAS NOT BEEN EXECUTED“.

BALTER erweitert bei Bedarf automatisch die Realms der bearbeiteten Datenbank. Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme).

BALTER berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

6.11.1 Analysephase

Mit Hilfe des alten und des neuen DBCOM bzw. der alten und der neuen SIA stellt BALTER in der Analysephase die Unterschiede fest zwischen der alten und der neuen Schema-Beschreibung und prüft, ob alle Änderungen erlaubt sind.

Am Ende der Analysephase protokolliert Ihnen BALTER wahlweise auf SYSLST (siehe REPORT-Anweisung, Tabelle 42, "Anweisungen für BALTER"), welche Benutzerrealms bei der Umstrukturierung hinzugefügt, gelöscht und benötigt werden, und die am Datenbestand auszuführenden Änderungen in der Reihenfolge, in der sie später in der Umstrukturierungsphase ablaufen.
Dieses Analyseprotokoll gibt Hinweise darauf, wie viel freien Speicherplatz BALTER für die einzelnen Umstrukturierungsaktionen in der Datenbank benötigt und ob Sie Änderungen planen, die nicht zugelassen sind.

Daher ist es wichtig, dass Sie dieses Analyseprotokoll ausgeben lassen und gut durchlesen und dass Sie die noch nicht gesicherten, benötigten Benutzterrealms sichern, bevor Sie die Umstrukturierungsphase starten.

> **i** Beim Anlegen eines eindeutigen Schlüssels (Sort-Key, CALC-Key, SEARCH KEY USING INDEX), der ausschließlich aus neu definierten Feldern besteht, für eine Satzart, zu der bereits Sätze in der Datenbank existieren, empfiehlt sich folgendes Vorgehen:
>
> 1. Neuen Schlüssel mit DUPLICATES ARE ALLOWED erzeugen
> 2. Den neuen Schlüssel mit eindeutigen Werten/Wertekombinationen belegen (z.B. mit Hilfe der DML-Anweisung MOVE)
> 3. Definition des Schlüssels ändern in DUPLICATES ARE NOT ALLOWED

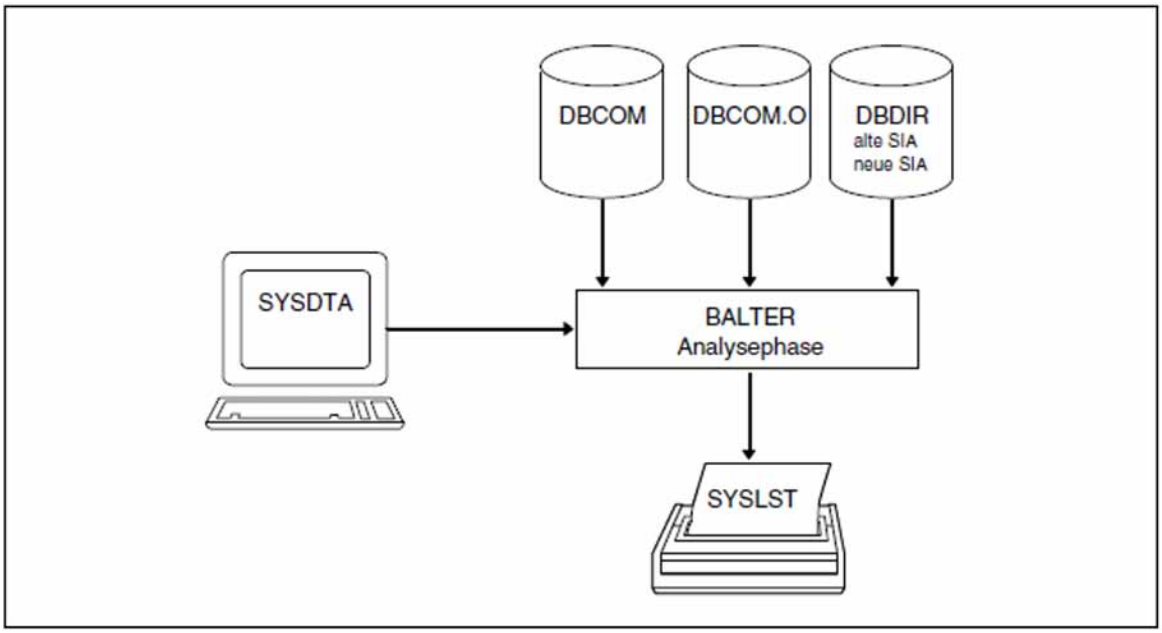

Bild 28: Systemumgebung in der Analysephase

Auf SYSOUT gibt BALTER Fehlermeldungen und Warnungen aus, wenn Sie Änderungen planen, die nicht erlaubt sind (siehe Abschnitt „Beschreibung der BALTER-Meldungen"). Bei REPORT IS YES gibt BALTER auf SYSOUT aus, welche Realms benötigt und welche Realms nicht benötigt werden.

6.11.2 Beschreibung des Analyseprotokolls (REPORT-Phase)

Geben Sie bei einem BALTER-Lauf REPORT IS YES an, so startet BALTER nach der Analysephase die REPORT-Phase, in der er das Analyseprotokoll auf SYSLST ausgibt. Im Analyseprotokoll listet BALTER die auszuführenden Änderungen in der Reihenfolge auf, in der er sie während der Umstrukturierungsphase durchführt.
Wenn BALTER für eine Änderung freien Speicherplatz in der Datenbank benötigt, so gibt er mit aus:

- die Anzahl der benötigten Datenseiten
- den Realm, in dem er den Platz benötigt

Außerdem protokolliert BALTER, ob und wie viel Speicherplatz bei den einzelnen Umstrukturierungsaktionen frei wird, und meldet Änderungen, die nicht oder nur unter bestimmten Bedingungen erlaubt sind.

> **Freier Speicherplatz**
> BALTER erweitert bei Bedarf automatisch die Realms der bearbeiteten Datenbank. Näheres hierzu siehe Handbuch „Datenbankbetrieb“, Automatische Realm-Erweiterung durch Dienstprogramme). Falls der freie Speicherplatz für eine Änderung nicht ausreicht, wird er in der Regel durch diese automatische Realm-Erweiterung zur Verfügung gestellt. Nur wenn die Voraussetzung für die automatische Realm-Erweiterung nicht erfüllt ist, kann der Fall eintreten, dass der freie Speicherplatz tatsächlich nicht ausreicht.

REPORT OF ADDED, DELETED AND NEEDED AREAS

Protokoll der hinzugefügten, gelöschten und benötigten Realms

Meldung	Bedeutung
realmname ADDED	Realm *realmname* hinzugefügt
realmname DELETED	Realm *realmname* gelöscht
realmname NEEDED	Realm *realmname* benötigt Realm *realmname* muss vor der Umstrukturierungsphase gesichert werden.

Tabelle 35: Protokoll der hinzugefügten, gelöschten und benötigten Realms

Bei REPORT IS YES protokolliert BALTER auch auf SYSOUT, welche Realms benötigt und welche Realms nicht benötigt werden

Meldung	Bedeutung
REALM NEEDED: *realmname*	Realm *realmname* benötigt Realm *realmname* muss vor der Umstrukturierungsphase gesichert werden.
REALM NOT NEEDED: *realmname*	Realm *realmname* nicht benötigt

Tabelle 36: SYSOUT-Protokoll der benötigten und nicht benötigten Realms

REPORT OF CHANGES IN DBTT FOR RECORD: satzname

Protokoll der DBTT-Änderungen für die Satzart *satzname*

Meldung	Bedeutung
`NUMBER OF ENTRIES RESERVED IN OLD DBTT PAGE = ganzzahl`	Anzahl der in einer Seite der bisherigen DBTT reservierten DBTT-Einträge
`NUMBER OF ENTRIES RESERVED IN NEW DBTT PAGE = ganzzahl`	Anzahl der in einer Seite der neuen DBTT reservierten DBTT-Einträge
`TOTAL NUMBER OF PAGES IN OLD DBTT = ganzzahl`	Anzahl der für die bisherige DBTT reservierten Seiten (DBTT-Basis und DBTT-Extents)
`TOTAL NUMBER OF PAGES, NEEDED FOR NEW DBTT IN AREA realmname = ganzzahl. IF THERE IS NOT ENOUGH SPACE AVAILABLE, THE RESTRUCTURING PROCESS WILL ABEND.`	Anzahl der Seiten, die die neue DBTT im Realm realmname insgesamt benötigt. Steht nicht ausreichend Platz zur Verfügung, so wird die Umstrukturierungsphase abgebrochen; für die neu anzulegenden DBTT-Extents müssen aufeinander folgende leere Seiten zur Verfügung stehen (vgl. **„Freier Speicherplatz"**).
`TABLE ANCHORED IN DBTT COLUMN: ganzzahl WILL BE DELETED.`	Die Tabelle, die in der DBTT-Spalte *ganzzahl* verankert war, wird gelöscht.
`TABLE ANCHORED IN DBTT-COLUMN-NR: ganzzahl WILL BE MOVED FROM {AREA realmname-1 | OWNER AREA} TO {AREA realmname-2 | OWNER AREA}.`	Die Tabelle, die in der DBTT-Spalte *ganzzahl* verankert ist, wird verlagert vom Realm *realmname-1* / Ownerrealm in den Realm *realmname-2* / Ownerrealm
`[UP TO]{ ganzzahl | NO} OLD DBTT PAGE(S) WILL BE FREED IN AREA realmname`	Anzahl der Seiten, die von der bisherigen DBTT im Realm *realmname* freigegeben werden. Die genaue Anzahl der tatsächlich freigegebenen Seiten in der Ausführungsphase kann in Einzelfällen um bis zu 64 Seiten geringer sein als in *ganzzahl* ausgewiesen.
`UP TO 256 CONSECUTIVE EMPTY PAM PAGES ARE NEEDED FOR NEW DBTT.`	Bei der Wiederverwendung von Teilen einer vorhandenen DBTT mit Extents wird eine neue DBTT-Basis angelegt, die größer als ein DBTT Extent ist.
`THE RECORD TYPE IS NOW OWNER IN SOME SETS.`	Durch die Umstrukturierung wird eine Satzart, die bisher nur Member in Sets war, auch zum Owner in Sets.

`THE DBTT HAS TO BE SHORTENED TO 16 711 679 ENTRIES.`	In Datenbanken mit 2-Kbyte-Seitenformat sind nur 16711679 DBTT-Einträge für Ownersatzarten möglich (vgl. Abschnitt „Mengengerüst beschreiben“ im Handbuch "Entwerfen und Definieren").
`IF THERE ARE EXISTING ENTRIES WITH HIGHER RSQ THE RESTRUCTURING PROCESS WILL END ABNORMALLY.`	Wenn in der bisherigen Membersatzart DBTT-Einträge oberhalb der für Ownersatzarten möglichen existieren, so wird BALTER die Umstrukturierungsphase abbrechen. Die vorgesehene Umstrukturierung ist im 2-Kbyte-Seitenformat nicht möglich.

Tabelle 37: Protokoll der DBTT-Änderungen

Belegt die neue DBTT mehr Seiten als die bisherige, so werden die bisher genutzten Seiten nach Möglichkeit weiter genutzt. Für neu anzulegende DBTT-Extents müssen aufeinander folgende leere Seiten mit der fixen Größe dieser Extents zur Verfügung stehen. Zusammenhängend müssen mindestens jeweils 128 freie PAM-Seiten für die DBTT-Teile zur Verfügung stehen.

Ist die neue DBTT genauso groß wie die bisherige oder kleiner, so verwendet BALTER zum Aufbauen der neuen DBTT die Seiten der bisherigen DBTT. Nicht mehr benötigte DBTT-Extents werden freigegeben.

REPORT OF DATABASE CHANGES FOR SINGULAR SET: setname

Protokoll der Änderungen für den singulären Set *setname*

Meldung	Bedeutung
`LENGTH OF OLD SYSTEM RECORD = ganzzahl`	Länge des alten Ankersatzes
`LENGTH OF NEW SYSTEM RECORD = ganzzahl`	Länge des neuen Ankersatzes

`TABLE OCCURRENCE ANCHORED IN SYSTEM RECORD`
`COLUMN ganzzahl`
`WILL BE DELETED IF PRESENT.`
	Die Tabelle, die in Spalte *ganzzahl* des Ankersatzes verankert ist, wird gelöscht, wenn sie existiert.

`TABLE ANCHORED IN DBTT-COLUMN-NR ganzzahl`
`WILL BE MOVED FROM`
`AREA realmname-1 TO AREA realmname-2.`
	Die Tabelle, die in Spalte *ganzzahl* des Ankersatzes verankert ist, wird vom Realm *realmname-1* in den Realm *realmname-2* verlagert.

`FORWARD CHAIN POINTER WILL BE REMOVED.`	Kettungsfeld für Vorwärtsverkettung wird entfernt
`BACKWARD CHAIN POINTER WILL BE REMOVED.`	Kettungsfeld für Rückwärtsverkettung wird entfernt
`THE SYSTEM RECORD WILL BE CREATED` `IN AREA realmname`	Der Ankersatz wird im Realm *realmname* aufgebaut.
`THE SYSTEM RECORD WILL BE DELETED` `IN AREA realmname`	Der Ankersatz wird aus dem Realm *realmname* gelöscht.
`THE SYSTEM RECORD WILL BE MOVED FROM` `AREA realmname-1 TO AREA realmname-2`	Der Ankersatz wird vom Realm *realmname-1* in den Realm *realmname-2* verlagert.

`ganzzahl PAGES WITH CALC SEARCH KEY TABLES`
`WILL BE FORMATED IN AREA realmname.`
`THEY ARE CONSECUTIVE.`
`IS THERE ENOUGH SPACE AVAILABLE?`
	Im Realm *realmname* werden *ganzzahl* Seiten für den indirekten Hashbereich des CALC-SEARCH-Keys formatiert.
	Die Seiten bilden einen zusammenhängenden Bereich. Reicht der freie Speicherplatz dafür?
	Vgl. **„Freier Speicherplatz"**.

`ganzzahl PAGES FOR CALC SEARCH KEY TABLES` `WILL BE DELETED IN AREA realmname`	Im Realm *realmname* werden *ganzzahl* CALC-Seiten von SEARCH-Key-Tabellen gelöscht.

Tabelle 38: Protokoll der Änderungen für singuläre Sets

REPORT OF DATABASE CHANGES FOR DELETION OF RECORD: satzname

Protokoll über das Löschen der Satzart *satzname*

Meldung	Bedeutung
`NUMBER OF ENTRIES RESERVED IN OLD DBTT PAGE =` *`ganzzahl`*	Anzahl der in einer Seite der bisherigen DBTT reservierten DBTT-Einträge
`ganzzahl` `OLD DBTT PAGES WILL BE FREED IN AREA` *`realmname`*	Im Realm *realmname* werden *ganzzahl* Seiten der bisherigen DBTT freigegeben.
`TABLE ANCHORED IN DBTT COLUMN` *`ganzzahl`* `WILL BE DELETED IF PRESENT.`	Die Tabelle, die in der DBTT-Spalte *ganzzahl* verankert war, wird gelöscht, wenn sie existiert.
`ganzzahl` `PAGES WITH CALC KEY RECORDS AND TABLES WILL BE DELETED IN AREA` *`realmname`*	Im Realm *realmname* werden *ganzzahl* direkte CALC-Seiten gelöscht.
`ganzzahl` `PAGES WITH CALC KEY TABLES WILL BE DELETED IN AREA` *`realmname`*	Im Realm *realmname* werden *ganzzahl* indirekte CALC-Seiten gelöscht.
`ALL RECORD INFORMATION WILL BE DELETED.`	Alle Sätze werden gelöscht.

Tabelle 39: Protokoll über das Löschen von Satzarten

REPORT OF DATABASE CHANGES FOR CREATION OF RECORD: satzname

Protokoll über das Hinzufügen der Satzart *satzname*

Meldung	Bedeutung
`NUMBER OF ENTRIES RESERVED IN NEW DBTT PAGE = `*`ganzzahl`	Anzahl der in einer Seite der neuen DBTT reservierten DBTT-Einträge
`TOTAL NUMBER OF PAGES, NEEDED FOR NEW DBTT IN AREA `*`realmname`* `= `*`ganzzahl`	Anzahl der leeren Seiten, die die neue DBTT im Realm *realmname* benötigt. Für die DBTT-Basis und die DBTT-Extents müssen aufeinander folgende leere Seiten zur Verfügung stehen.
`ganzzahl` `PAGES FOR CALC KEY RECORDS AND TABLES WILL BE FORMATED IN AREA `*`realmname`*`. THEY ARE CONSECUTIVE.`	Im Realm *realmname* werden *ganzzahl* Seiten für einen direkten Hashbereich formatiert. Sie bilden einen zusammenhängenden Bereich.
`ganzzahl` `PAGES FOR CALC KEY TABLES WILL BE FORMATED IN AREA `*`realmname`*`. THEY ARE CONSECUTIVE.`	Im Realm *realmname* werden *ganzzahl* Seiten für einen indirekten Hashbereich formatiert. Sie bilden einen zusammenhängenden Bereich.
`IF THERE IS NOT ENOUGH SPACE AVAILABLE, THE RESTRUCTURING PROCESS WILL END ABNORMALLY.`	Steht nicht genügend Platz zur Verfügung, so bricht BALTER die Umstrukturierungsphase ab. Vgl. **„Freier Speicherplatz"**.

Tabelle 40: Protokoll über das Hinzufügen von Satzarten

REPORT OF DATABASE CHANGES FOR RECORD: satzname

Protokoll der Änderungen für die Satzart *satzname*

Meldung	Bedeutung
ganzzahl PAGES WITH CALC KEY RECORDS AND TABLES WILL BE DELETED IN AREA *realmname*	Im Realm *realmname* werden *ganzzahl* direkte CALC-Seiten gelöscht.
ganzzahl PAGES WITH CALC KEY TABLES WILL BE DELETED IN AREA *realmname*	Im Realm *realmname* werden *ganzzahl* indirekte CALC-Seiten gelöscht.
ganzzahl PAGES FOR CALC KEY RECORDS AND TABLES WILL BE FORMATED IN AREA *realmname*. THEY ARE CONSECUTIVE. IS THERE ENOUGH SPACE AVAILABLE?	Im Realm *realmname* werden *ganzzahl* Seiten für einen direkten Hashbereich formatiert. Sie bilden einen zusammenhängenden Bereich. Reicht der freie Speicherplatz dafür? Vgl. **„Freier Speicherplatz"**.
ganzzahl PAGES FOR CALC KEY TABLES WILL BE FORMATED IN AREA *realmname*. THEY ARE CONSECUTIVE. IS THERE ENOUGH SPACE AVAILABLE?	Im Realm *realmname* werden *ganzzahl* Seiten für einen indirekten Hashbereich formatiert. Sie bilden einen zusammenhängenden Bereich. Reicht der freie Speicherplatz dafür? Vgl. **„Freier Speicherplatz"**.
AREA DELETED FROM RECORD-WITHIN-CLAUSE	Realm *realmname* aus der RECORD-WITHIN-Klausel entfernt
FOLLOWING ACTIONS EXECUTED IF RECORD OCCURRENCES ARE PRESENT:	Sind Sätze der Satzart gespeichert, so wird Folgendes durchgeführt:
THE RESTRUCTURING PROCESS WILL END ABNORMALLY FOR NOT ALLOWED SCHEMA CHANGES.	Wegen nicht erlaubter Schemaänderungen wird BALTER die Umstrukturierungsphase abbrechen.
THE RESTRUCTURING PROCESS WILL END ABNORMALLY IF RECORD OCCURRENCES ARE PRESENT IN AREAS WHICH ARE DELETED FROM RECORD-WITHIN-CLAUSE	Sind in den Realms, die aus der RECORD-WITHIN-Klausel entfernt wurden, Sätze der Satzart gespeichert, so wird BALTER die Umstrukturierungsphase abbrechen.
A NON SINGULAR AUTOMATIC SET THAT WAS NOT PRESENT IN THE OLD SCHEMA HAS BEEN SPECIFIED IN THE NEW SCHEMA. THE RESTRUCTURING PROCESS WILL STOP BECAUSE THE SET OCCURRENCES TO WHICH EACH RECORD OCCURRENCE BELONGS ARE NOT KNOWN.	Sie haben im neuen Schema einen standard Set mit Set-Mitgliedschaft MANDATORY AUTOMATIC hinzugefügt. BALTER wird die Umstrukturierungsphase abbrechen, da die Set-Occurrences, denen die einzelnen Sätze zugeordnet werden sollen, nicht bekannt sind.

AS A CONSEQUENCE OF LOGICAL CHANGE THE RECORDTYPE WILL BE PLACED IN NEW PAGES. DURING THE PROCESS THE RECORDTYPE WILL RESIDE TWICE IN THE AREA(S). IS THERE ENOUGH SPACE AVAILABLE?	Wegen vollständiger Verlagerung wird die Satzart in andere Seiten verlagert. Während des Umspeicherns wird dieSatzart in dem (den) Realm(s) doppeltvorkommen. Reicht der freie Speicherplatz dafür? Vgl. **„Freier Speicherplatz"**.
SET *setname* DOES NOT HAVE MODE = LIST ANYMORE. THE LIST TABLE HEADER WILL BE REMOVED FROM THE LIST-PAGES.	Der Set *setname* ist nicht mehr mit MODE IS LIST definiert. BALTER wird den Tabellenkopf der Liste aus den Seiten der Liste entfernen.
DUE TO A CHANGE IN LOCATION MODE THE CALC KEY INFORMATION WILL BE REMOVED.	Die CALC-Information wird auf Grund einer Änderung der LOCATION MODE-Klausel entfernt.
CALC RECORDS AND KEYS WILL BE PLACED IN THE CALC KEY PAGES.	BALTER wird CALC-Sätze und -Keys in den Seiten des Hashbereichs eintragen.
CALC KEYS WILL BE PLACED IN THE CALC KEY PAGES.	BALTER wird CALC-Keys in den Seiten des Hashbereichs eintragen.
'DUPLICATES ARE NOT ALLOWED' HAS BEEN SPECIFIED FOR THE CALC KEY. IF DUPLICATES ARE DETECTED THE DUPLICATE VALUES WILL BE PRINTED AND THE RESTRUCTURING PROCESS WILL CONTINUE.	DUPLICATES ARE NOT ALLOWED wurde für den CALC-Key festgelegt. Findet BALTER Duplikate, so protokolliert er die Duplikatwerte und setzt die Umstrukturierungsphase fort. Enthält die Tabelle nach dem BALTER-Lauf Duplikate, so kann die Tabelle über diese Duplikate nicht verarbeitet werden. Dies lässt sich wie folgt korrigieren: Zunächst DUPLICATES ARE ALLOWED in der Schema-DDL definieren, danach die Schlüsselfelder gemäß DUPLICATES NOT belegen und anschließend die DDL-Definition in DUPLICATES NOT ändern.

'DUPLICATES ARE NOT ALLOWED' HAS BEEN SPECIFIED FOR THE CALC KEY. THIS KEY IS A NEW ONE ON {ONE NEW FIELD | ONLY NEW FIELDS}. THEREFORE THE TABLE WILL HAVE ONLY DUPLICATES. THIS IS INCONSISTENT WITH 'DUPLICATES ARE NOT ALLOWED'. SPECIFY 'DUPLICATES ARE ALLOWED' IF DUPLICATES ARE DETECTED THE DUPLICATE VALUES WILL BE PRINTED AND THE RESTRUCTURING PROCESS WILL CONTINUE.	DUPLICATES ARE NOT ALLOWED wurde für den CALC-Key festgelegt. Jedes Feld dieses Schlüssels ist neu. Werden vorhandene Sätze der Satzart, für die der Schlüssel definiert ist, zum Tabellenaufbau benutzt, so wird eine Tabelle aus sämtlich neuen Feldern aufgebaut und kann nur Duplikate enthalten. BALTER protokolliert die gefundenen Duplikate und setzt die Umstrukturierungsphase fort. Die Tabelle kann jedoch nicht verarbeitet werden. Die widersprüchlichen Vorgaben lassen sich wie folgt korrigieren: Zunächst DUPLICATES ARE ALLOWED in der Schema-DDL definieren, danach die neuen Schlüsselfelder gemäß DUPLICATES NOT belegen und anschließend die DDL-Definition in DUPLICATES NOT ändern.
FOR SET *setname* A SORTED CHAIN WILL BE BUILT.	Für den Set *setname* wird eine sortierte Kette aufgebaut.
FOR SET *setname* A LIST TABLE WILL BE BUILT.	Für den Set *setname* wird eine Liste aufgebaut.
FOR SET *setname* A POINTER ARRAY WILL BE BUILT.	Für den Set *setname* wird eine Adressliste aufgebaut.
'DUPLICATES ARE NOT ALLOWED' HAS BEEN SPECIFIED FOR THE SORT KEY. IF DUPLICATES ARE DETECTED THE DUPLICATE VALUES WILL BE PRINTED AND THE RESTRUCTURING PROCESS WILL CONTINUE.	DUPLICATES ARE NOT ALLOWED wurde für den ASC-/DESC-Key festgelegt. Findet BALTER Duplikate, so protokolliert er die Duplikatwerte und setzt die Umstrukturierungsphase fort. Enthält die Tabelle nach dem BALTER-Lauf Duplikate, so kann die Tabelle über diese Duplikate nicht verarbeitet werden. Dies lässt sich wie folgt korrigieren: Zunächst DUPLICATES ARE ALLOWED in der Schema-DDL definieren, danach die Schlüsselfelder gemäß DUPLICATES NOT belegen und anschließend die DDL-Definition in DUPLICATES NOT ändern.

'DUPLICATES ARE NOT ALLOWED' HAS BEEN SPECIFIED FOR THE SORT KEY. THIS KEY IS A NEW ONE ON {ONE NEW FIELD | ONLY NEW FIELDS}. THEREFORE THE TABLE WILL HAVE ONLY DUPLICATES. THIS IS INCONSISTENT WITH 'DUPLICATES ARE NOT ALLOWED'. SPECIFY 'DUPLICATES ARE ALLOWED' IF DUPLICATES ARE DETECTED THE DUPLICATE VALUES WILL BE PRINTED AND THE RESTRUCTURING PROCESS WILL CONTINUE.	DUPLICATES ARE NOT ALLOWED wurde für den ASC-/DESC-Key festgelegt. Jedes Feld dieses Schlüssels ist neu. Werden vorhandene Sätze der Satzart, für die der Schlüssel definiert ist, zum Tabellenaufbau benutzt, so wird eine Tabelle ausschließlich aus neuen Feldern aufgebaut und kann nur Duplikate enthalten. BALTER protokolliert die gefundenen Duplikate und setzt die Umstrukturierungsphase fort. Die Tabelle kann jedoch nicht verarbeitet werden. Die widersprüchlichen Vorgaben lassen sich wie folgt korrigieren: Zunächst DUPLICATES ARE ALLOWED in der Schema-DDL definieren, danach die neuen Schlüsselfelder gemäß DUPLICATES NOT belegen und anschließend die DDL-Definition in DUPLICATES NOT ändern.
FOR SET *setname* CALC SEARCH KEYS WILL BE PLACED IN THE CALC KEY PAGES.	Für den Set *setname* werden CALC-SEARCH-Keys in die Seiten des indirekten Hashbereichs eingetragen.
FOR SET *setname* AN INDEXED SEARCH KEY TABLE OF TYPE REPEATED KEY WILL BE BUILT.	Für den Set *setname* wird eine mehrstufige SEARCH-Key-Tabelle des Typs REPEATED KEY aufgebaut.
FOR SET *setname* AN INDEXED SEARCH KEY TABLE OF TYPE DATABASE KEY LIST WILL BE BUILT.	Für den Set *setname* wird eine mehrstufige SEARCH-Key-Tabelle des Typs DATABASE-KEY-LIST aufgebaut.

'DUPLICATES ARE NOT ALLOWED' HAS BEEN SPECIFIED FOR THE SEARCH KEY. IF DUPLICATES ARE DETECTED THE DUPLICATE VALUES WILL BE PRINTED AND THE RESTRUCTURING PROCESS WILL CONTINUE.	DUPLICATES ARE NOT ALLOWED wurde für den SEARCH-Key festgelegt. Findet BALTER Duplikate, so protokolliert er die Duplikatwerte und setzt die Umstrukturierungsphase fort. Enthält die Tabelle nach dem BALTER-Lauf Duplikate, so kann die Tabelle über diese Duplikate nicht verarbeitet werden. Dies lässt sich wie folgt korrigieren: Zunächst DUPLICATES ARE ALLOWED in der Schema-DDL definieren, danach die Schlüsselfelder gemäß DUPLICATES NOT belegen und anschließend die DDL-Definition in DUPLICATES NOT ändern.
'DUPLICATES ARE NOT ALLOWED' HAS BEEN SPECIFIED FOR THE SEARCH KEY. THIS KEY IS A NEW ONE ON {ONE NEW FIELD | ONLY NEW FIELDS}. THEREFORE THE TABLE WILL HAVE ONLY DUPLICATES. THIS IS INCONSISTENT WITH 'DUPLICATES ARE NOT ALLOWED'. SPECIFY 'DUPLICATES ARE ALLOWED' IF DUPLICATES ARE DETECTED THE DUPLICATE VALUES WILL BE PRINTED AND THE RESTRUCTURING PROCESS WILL CONTINUE.	DUPLICATES ARE NOT ALLOWED wurde für den SEARCH-Key festgelegt. Jedes Feld dieses Schlüssels ist neu. Werden vorhandene Sätze der Satzart, für die der Schlüssel definiert ist, zum Tabellenaufbau benutzt, so wird eine Tabelle ausschließlich aus neuen Feldern aufgebaut und kann nur Duplikate enthalten. BALTER protokolliert die gefundenen Duplikate und setzt die Umstrukturierungsphase fort. Die Tabelle kann jedoch nicht verarbeitet werden. Die widersprüchlichen Vorgaben lassen sich wie folgt korrigieren: Zunächst DUPLICATES ARE ALLOWED in der Schema-DDL definieren, danach die neuen Schlüsselfelder gemäß DUPLICATES NOT belegen und anschließend die DDL-Definition in DUPLICATES NOT ändern.
ALL RECORD OCCURRENCES WILL BE READ AND WRITTEN.	Alle Sätze werden gelesen und geschrieben.
ALL RECORD OCCURRENCES WILL BE READ, MODIFIED AND WRITTEN. THESE MODIFICATIONS ARE A CONSEQUENCE OF CHANGES IN: - THE SYSTEM-PART OF THE RECORD - THE USER-PART OF THE RECORD	Alle Sätze werden gelesen, geändert und geschrieben. Dies ist notwendig wegen Änderungen in den Set-Connection-Data der Satzart und im Benutzerteil der Satzart

`THE SYSTEM WILL TRY TO USE SAME PAGE FOR THE NEW ALLOCATION OF THE RECORD OCCURRENCES.`	BALTER wird versuchen, den Sätzen beim Neu-Speichern die alten Seiten zuzuweisen.
`LIST WILL BE REALLOCATED`	Eine Liste wird neu aufgebaut. Membersätze eines verteilbaren SYSTEM-LIST-Set werden annähernd gleich über die beteiligten Realms verteilt.
`RECORDS OF SYSTEM LIST SET CAN NOW BE STORED IN n REALMS`	Nach der Umstrukturierung können die Membersätze in *n* Realms abgespeichert werden.

Tabelle 41: Protokoll der Änderungen von Satzarten

Falls Sie den Benutzerteil einer Satzart geändert haben, gibt BALTER eine Tabelle aus, die das Layout des alten Satzes dem neuen gegenüberstellt:

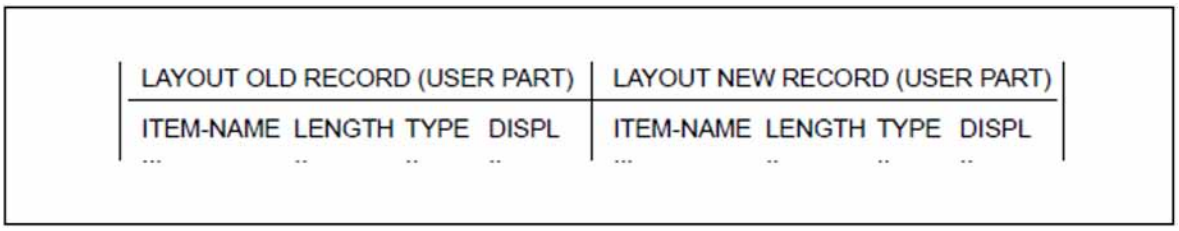

Bild 29: Vergleich des alten und des neuen Satzes (Benutzerteil)

LAYOUT OLD RECORD (USER PART)

altes Layout der Satzart (Benutzerteil)

LAYOUT NEW RECORD (USER PART)

neues Layout der Satzart (Benutzerteil)

ITEM-NAME

Feldname

TYPE Typ des Feldes:

1 alphanumerische Zeichenfolge
2 entpackte Dezimalzahl ohne Vorzeichen
3 entpackte Dezimalzahl mit Vorzeichen
5 gepackte Dezimalzahl
6 Halbwort
7 Wort
8 Database-Key-Feld

DISPL Distanz des Feldes zum Beginn der Satzart (incl. SCD)

Maßnahmen bei Speicherbedarf in den Realms

Wenn Sie anhand des Analyseprotokolls feststellen, dass für die Umstrukturierung in einem oder mehreren Realms Ihrer Datenbank mehr Speicherplatz benötigt wird, als frei ist, so müssen Sie folgende Maßnahmen durchführen:

- Entweder die Voraussetzungen für eine automatische Erweiterbarkeit der betroffenen Realms schaffen (Näheres hierzu siehe Handbuch „Datenbankbetrieb“)
- oder manuell zusätzlichen Speicherplatz in den betroffenen Realms schaffen:
  - die bisher veränderten Realms Ihrer Datenbank zurücksetzen auf den Stand vor Beginn der Umstrukturierung (siehe Abschnitt „Datenbank rekonstruieren")
  - mit BREORG die betroffenen Realms erweitern
  - den Umstrukturierungsprozess ab ’Vorbereiten der Compilerdatenbank’ neu starten.

6.11.3 Umstrukturierungsphase

Die Umstrukturierungsphase startet BALTER dann, wenn Sie sie per Steueranweisung anfordern (siehe EXECUTION-Anweisung, Tabelle 42, "Anweisungen für BALTER") und die Analysephase keine Fehler feststellt.

6.11.3.1 Auswirkungen der Umstrukturierung auf den Inhalt der Datenbank

Die verschiedenen Änderungen, die Sie am Schema und an der Speicherstruktur vornehmen können, beeinflussen den Inhalt der Datenbank sehr unterschiedlich:

- BALTER ändert in der Umstrukturierungsphase den Inhalt der Datenbank in folgenden Fällen nicht:
  - wenn Sie *bezeichner* umbenennen
  - wenn Sie LOCATION DIRECT bzw. DIRECT-LONG hinzufügen oder weglassen
  - wenn Sie in der ORDER-Klausel variieren zwischen: LAST, FIRST, NEXT, PRIOR oder IMMATERIAL
  - wenn Sie die Set-Mitgliedschaft neu festlegen
  - wenn Sie Duplikate erlauben oder verbieten
  - wenn Sie die SET OCCURRENCE SELECTION-Klausel ändern

  Solche Änderungen vermerkt BALTER nur in den Datenbankdefinitionen; sie wirken sich nicht auf den Datenbestand aus, sondern sind lediglich beim Programmieren der DB-Anwendungen zu berücksichtigen.
- BALTER ändert den Inhalt der Datenbank in folgenden Fällen ebenfalls nicht:
  - wenn Sie die Größe einer DBTT neu festlegen
  - wenn Sie die Seitenanzahl für das dynamische Reorganisieren von Tabellen neu festlegen
  - wenn Sie die Größe von Set-Tabellen neu festlegen

  Derartige Änderungen müssen Sie mit dem Dienstprogramm BREORG durchführen.
- BALTER ändert den Inhalt der Datenbank in folgenden Fällen nur bedingt:
  - wenn Sie PLACEMENT OPTIMIZATION neu festlegen
  - wenn Sie die Lage von Tabellen neu festlegen
  - wenn Sie die Größe eines Hashbereichs neu festlegen

  Derartige Änderungen beeinflussen zwar den Inhalt der Datenbank, ihre Konsistenz hängt aber nicht davon ab, ob BALTER den Datenbestand an die neue Definition anpasst oder nicht. Da BALTER außerdem bei solchen Änderungen in den meisten Fällen sehr viele Sätze bzw. Tabellenzeilen umspeichern muss, aktualisiert er bei diesen Änderungen zwar generell die Datenbankdefinition, passt aber den Datenbestand nur dann an die neue Definition an, wenn er die Sätze bzw. Tabellen auf Grund weiterer Änderungen verlagern muss.
- Generell ändert BALTER den Inhalt der Datenbank in folgenden Fällen:
  - wenn Sie einen Realm zur Datenbank hinzufügen
  - wenn Sie eine neue Satzart oder einen neuen Set definieren
  - wenn Sie LOCATION CALC erstmalig definieren
  - wenn Sie bei LOCATION CALC die Hashroutine oder die CALC-Keys ändern
  - wenn Sie eine DBTT in einen anderen Realm verlagern
  - wenn Sie die Definition eines SEARCH-Keys ändern
  - wenn Sie die Benutzer- oder Systeminformation einer Satzart ändern
  - wenn Sie den Aufbau oder die Sortierkriterien von Tabellen neu festlegen

  BALTER passt bei diesen Änderungen den Datenbestand an das geänderte Schema bzw. die geänderte Speicherstruktur an, indem er folgende Aktionen in der genannten Reihenfolge ausführt:
  - hinzugefügte Realms formatiert
  - bei Ownersatzarten die DBTT modifiziert und die Set-Tabellen löscht oder in der Datenbank verlagert
  - Ankersätze für SYSTEM-Sets einträgt, löscht oder modifiziert

- Tabellen löscht oder verlagert
- alle Informationen von Satzarten löscht, die im neuen Schema nicht mehr vorkommen
- Hashbereiche und DBTTs für neue Satzarten anlegt
- Satzarten (Benutzer- und Systeminformation) modifiziert und umspeichert und Tabellen neu aufbaut
- Realms, die im neuen Schema nicht mehr vorkommen, von der Datenbank abhängt

> **i** Wurde mit dem Dienstprogramm BMODTT das Wiederverwenden von Database Keys ausgeschaltet, so muss nach dem Umstrukturieren dieser Lauf wiederholt werden, da BGSIA die Standard-Einstellung (Database Keys gelöschter Sätze sind wiederverwendbar bzw. Freiplatzsuche beginnt in einem zusammenhängend freien Bereich am Realm-Ende) wiederherstellt. Löschkennzeichen bleiben bei Neuaufbau von DBTTs durch BALTER erhalten.

6.11.3.2 Protokollierung der Umstrukturierungsphase

Alle Aktionen der Umstrukturierungsphase können Sie dem Analyseprotokoll entnehmen. Während der Umstrukturierungsphase protokolliert BALTER nur Folgendes auf SYSOUT:

- das Hinzufügen und Löschen von Realms oder Satzarten
- das Verarbeiten von Satzarten, d.h. Aufbau neuer Tabellen und Modifizieren von Satzarten

6.11.3.3 Systemumgebung in der Umstrukturierungsphase

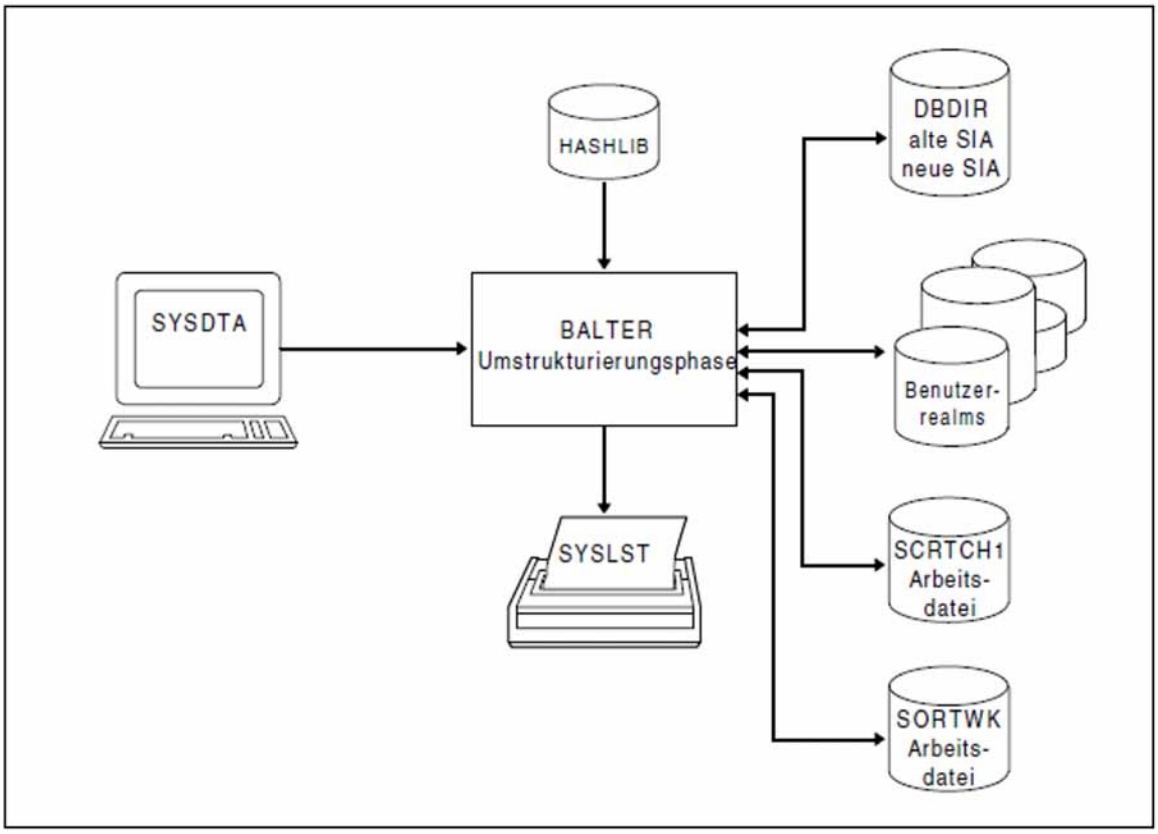

Bild 30: Systemumgebung in der Umstrukturierungsphase

Realms

- Benutzerrealms
  Welche Benutzerrealms BALTER in der Umstrukturierungsphase benötigt, kann durch einen Analyselauf mit REPORT IS YES EXECUTION IS NO abgefragt werden.Dabei wird auf die Benutzerrealms noch nicht zugegriffen. Erst in der Umstrukturierungsphase greift BALTER auf die benötigten Benutzerrealms zu.
- hinzugefügte Realms
  Wenn es sich nicht um Temporäre Realms handelt, müssen Sie die hinzugefügten Realms vor dem Starten der Umstrukturierungsphase per CREATE-FILE-Kommando anlegen mit dem Dateinamen:
  dbname.realmname

Arbeitsdateien

BALTER benötigt während der Umstrukturierungsphase auf Platte zwei Arbeitsdateien, die er automatisch unter der entsprechenden Benutzerkennung auf gemeinschaftlicher Platte anlegt und nach normaler Beendigung des Laufs wieder löscht.

Die Dateien haben die Standard-Dateikettungsnamen SCRTCH1 und SORTWK:

SCRTCH1	benötigt BALTER zum Zwischenspeichern von Informationen für das Umspeichern und Modifizieren von Sätzen und das Aufbauen von Tabellen
SORTWK	benötigt der von BALTER benutzte SORT für die Sortierung interner Auswertungssätze (siehe Handbuch **„SORT (BS2000)"**).

Wollen Sie die beiden Arbeitsdateien explizit einrichten, so müssen diese die folgenden Eigenschaften besitzen:

Arbeitsdatei-1

Die Primärzuweisung für die Arbeitsdatei-1 sollte sich am Mengengerüst der zwischenzuspeichernden Daten orientieren. Es sollte immer eine angemessene Sekundärzuweisung erfolgen für den Fall, dass der Speicherplatz erweitert werden muss.

Dateikettungsname SCRTCH1

Zugriffsmethode PAM

Das Mengengerüst der zwischenzuspeichernden Daten ergibt sich annähernd aus der folgenden Formel:

`max(`*`schlüssellänge`* `x` *`anzahl sätze`*`) x 3 Bytes`

schlüssellänge

Gesamtlänge aller benötigten Schlüssel. Als Minimum ist 8 zu wählen.

max

höchster Wert, der sich beim Verarbeiten verschiedener Satzarten ergibt

Arbeitsdatei-2

Die Arbeitsdatei-2 wird vom SORT benötigt, wenn der virtuelle Speicher für die Vorsortierung nicht ausreicht. Die Primärzuweisung sollte sich am Mengengerüst der zu sortierenden Daten orientieren unter Berücksichtigung des von SORT empfohlenen Sicherheitsfaktors (siehe Abschnitt „Arbeitsdateien“ im Handbuch **„SORT (BS2000)"**). Es sollte immer eine angemessene Sekundärzuweisung erfolgen für den Fall, dass der Speicherplatz erweitert werden muss.

Dateikettungsname SORTWK

Zugriffsmethode PAM

Das Mengengerüst der zu sortierenden Daten ergibt sich aus der folgenden Formel:

`max(`*`satzlänge`* `x` *`anzahl sätze`*`) Bytes`

satzlänge Länge eines Satzes inkl. SCD

max höchster Wert, der sich beim Verarbeiten verschiedener Satzarten ergibt

Richten Sie die beiden Arbeitsdateien nicht selbst ein, so richtet BALTER sie mit folgenden Namen und Größen ein:

UTI.*tsn*.SCRTCH1 (360,360)

UTI.*tsn*.SORTWK (120,120)

tsn ist die Prozessfolgenummer, unter der Sie BALTER starten.

6.11.4 Anweisungen für BALTER

BALTER kennt die folgenden Anweisungen:

Anweisung	Standardwert	Bedeutung
[SORTCORE IS *nnn*.]	150	Größe des Sortierpuffers festlegen
EXECUTION IS { YES | NO }.	-	Umstrukturierungsphase starten/nicht starten
REPORT IS { YES | NO }.	-	Protokollierung anfordern/unterbinden
[FILL *feld name* OF *satz name* WITH *wert* .]	-	Initialisiert den Feldnamen *feldname* von *satzname* in allen existierenden Sätzen mit dem benutzerdefinierten Wert anstatt mit Nullen oder Leerzeichen.
[FILLING IS *nnn* PERCENT [IN SET NAME IS {*setname*,... | *ALL[EXCEPT *setname*,..]}].]		Füllgrad von Tabellen festlegen (Format 1)
[FILLING WITH POPULATION [IN SET NAME IS {*setname*,... | *ALL[EXCEPT *setname*,..]}].]		Füllgrad von Tabellen festlegen (Format 2)
END .	-	beendet die Eingabe der Anweisungen

Table 42: Anweisungen für BALTER

Die Anweisungen werden nachfolgend im Detail beschrieben.

SORTCORE (Größe des Sortierpuffers festlegen)

Zum Sortieren von Elementen (Datensätze/Tabellenzeilen) verwendet BALTER das BS2000-Dienstprogramm SORT. Mit der Anweisung SORTCORE bestimmen Sie die Größe des Hauptspeicherplatzes für den von SORT genutzten Sortierbereich (siehe Handbuch **„SORT (BS2000)"** , ALLOC-Anweisung).

```
[SORTCORE IS nnn.]
```

nnn Sie legen die Größe des Speicherplatzes in Einheiten von 4-Kbyte für den Sortierpuffer fest, der dem Dienstprogramm SORT des BS2000 zur Verfügung gestellt wird (siehe Handbuch **„SORT (BS2000)"** , ALLOC-Anweisung).
Standardwert:150

Das Mengengerüst der zu sortierenden Daten ist das gleiche, das der Größe der Arbeitsdatei-2 zugrunde liegt (siehe "Systemumgebung in der Umstrukturierungsphase").

EXECUTION (Umstrukturierungsphase starten/nicht starten)

Mit der Anweisung EXECUTION legen Sie fest, ob BALTER die Änderungen am Schema und an der Speicherstruktur nur in der Analysephase analysieren oder auch bereits den Datenbestand in der Umstrukturierungsphase an die Änderungen anpassen soll.

Diese Anweisung müssen Sie angeben.

```
EXECUTION IS {YES | NO}.
```

NO	nur Analysephase
YES	Analysephase und Umstrukturierungsphase

REPORT (Protokollierung anfordern/unterbinden)

Mit der Anweisung REPORT legen Sie fest, ob BALTER ein Analyseprotokoll ausdrucken soll oder nicht (s. Abschnitt Beschreibung des Analyseprotokolls (REPORT-Phase)).

Diese Anweisung müssen Sie angeben.

```
REPORT IS {YES | NO}.
```

YES	BALTER druckt ein Analyseprotokoll aus
NO	BALTER protokolliert nichts

FILL (Felder mit benutzerdefiniertem Wert initialisieren)

Die FILL-Anweisung kann verwendet werden, um neue Felder mit einem benutzerdefinierten Wert zu initialisieren anstatt mit Nullen oder Leerzeichen.

```
[FILL feldname OF satzname WITH wert.]
```

Dabei sind *feldname* und *satzname* Bezeichner für den Satz und das Feld in dem Satz. Bei *feldname* muss es sich um ein neues Feld handeln.

Folgende Werte sind möglich:

- ***[+,-]n[,n]*** für numerische Werte, wobei n eine Folge von Dezimalzahlen (0-9) ist. Als Trennzeichen für den Integer-Teil und den Nachkomma-Teil wird das Komma (‘,’) verwendet.
- ***‘Z’*** für alphanumerische Werte, wobei Z eine Folge von Characterzeichen ist.
- ***X’n’*** für alle Typen; der Wert wird in Hexadezimaldarstellung eingegeben, wobei n eine gerade Anzahl vonbvHexadezimalzeichen (0-F) ist.

Hinweise für Binärtypen und ungepackte Typen:

Der Wert darf nur ‘+’, ‘-’, Komma und Zahlen von 0 bis 9 enthalten und der ersten Ziffer muss ‘+’ bzw. ‘-’ vorangestellt sein. Die Ziffern müssen rechtsbündig angeordnet sein. Ist dem Wert kein Vorzeichen vorangestellt, so hat er einen positiven Wert. Der Wert darf nicht mehr als ein Komma enthalten.

Hinweise für Datenbankschlüssel:

Der Wert darf nur im Binärformat in Hexadezimaldarstellung eingegeben werden.

Beispiel 1:

Die Datenbank hat die folgende Struktur (nach der Restrukturierung):

```
...
RECORD NAME IS      EMPLOYEE .
01 ID1              TYPE IS DATABASE-KEY-LONG.
01 FIRSTNAME        TYPE IS CHARACTER 6.
01 SALARY           TYPE IS FIXED REAL DECIMAL 7,3.
01 DEBT             TYPE IS BINARY 31.

RECORD NAME IS      IMPORTANT-DATA .
01 ID2              TYPE IS DATABASE-KEY-LONG.
01 DATA             PICTURE IS X(200).
...
```

Kommando zum Starten von BALTER

```
01  /START-UDS-BALTER
02  REPORT IS YES.
03  EXECUTION IS YES.
04  FILL FIRSTNAME OF EMPLOYEE WITH ‘JOHN’.
05  FILL SALARY OF EMPLOYEE WITH 1535,89.
06  FILL DEBT OF EMPLOYEE WITH -120000.
07  FILL DATA OF IMPORTANT-DATA WITH X’A1B2C3’.
08  END
```

Der Wert wird in den Datentyp des Feldes feldname konvertiert. Ist dies nicht möglich, so wird die Ausführung von BALTER beendet.
Die maximale Länge des eingegebenen Wertes beträgt 255 Zeichen.

Falls der Wert so lang ist, dass die Länge der FILL Anweisung länger als 72 Zeichen wird, so können Sie die Anweisungen wie im nachfolgenden Beispiel 2 angeben.
BALTER liest jeweils 72 Zeilen bis ein abschliessendes Hochkomma gefunden wird.

Beispiel 2:

FILL DATA OF IMPORTANT-DATA WITH 'DataDataDataDataDataDataDataDataDataDa

taDataDataDataDataDataDataDataDataDataDataDataDataDataDataDataDataDataDa

taData'

Beispiel 3**:**

Wenn die Datenbank die gleiche Struktur hat und die FILL-Anweisung wie im obigen Beispiel eingegeben wird, dann werden alle Datensätze folgendermaßen aufgefüllt:

FIRSTNAME	‘JOHN ’
SALARY	+1535,890
DEBT	-120 000
DATA	X’A1B2C3’

Wenn ungültige Zeichen für den Wert eingegeben werden oder der Wert nicht in den zugehörigen Typ konvertiert werden kann, dann wird die Ausführung von BALTER beendet und eine Fehlermeldung ausgegeben.

Es wird empfohlen, BALTER zuerst mit der Anweisung ”EXECUTION IS NO” auszuführen und die Fehlermeldungen zu analysieren.

FILLING (Füllgrad von Tabellenseiten festlegen)

Mit der Anweisung FILLING legen Sie den Füllgrad neu aufgebauter Tabellen fest.

Format 1	legt einen prozentualen Füllgrad der neu aufgebauten Tabellen fest.
Format 2	legt eine Mindestgröße für neu aufgebaute Tabellen fest. Die tatsächlich genutzte Mindestgröße wird jedoch durch die Anzahl der Tabelleneinträge begrenzt, die in eine Datenbankseite passen. Deshalb wirkt Format 2 nur auf kleine Tabellen, also solche die in eine Seite passen.

Format 1

```
[FILLING IS nnn PERCENT
[ IN SET NAME IS {setname,... | *ALL[ EXCEPT setname,..]}].]
```

Format 2

```
[FILLING WITH POPULATION
[ IN SET NAME IS {setname,... | *ALL[ EXCEPT setname,..]}].]
```

nnn

legt fest, zu wie viel Prozent neu aufzubauende Tabellenseiten gefüllt werden

nnn = 1 ... 100

POPULATION

legt fest, dass zur Bestimmung der Größe neu aufzubauender Tabellen die POPULATION-Klausel in der SSL herangezogen wird.

IN SET NAME IS ...

legt fest, in welchen Sets für neu aufzubauende Tabellen angegebene Füllgrad gilt.Wenn Sie IN SET NAME IS weglassen, wirkt FILLING auf alle neu aufzubauenden Tabellenseiten.

setname,...

der angegebene Füllgrad gilt für neu aufzubauende Tabellen der angegebenen Sets.

*ALL

Der angegebene Füllgrad gilt für alle neu aufzubauenden Tabellen.

*ALL EXCEPT *setname*,...

Der angegebene Füllgrad gilt für alle neu aufzubauenden Tabellen, außer in den nach EXCEPT angegebenen Sets.

- Format 1 der Anweisung wirkt auf einstufige Tabellen und auf die Stufe 0 aller mehrstufigen Tabellen, die BALTER neu aufbaut, außer auf Listen.
  Der Standard-Füllgrad für Tabellenseiten auf Stufe 1 beträgt 95%, auf allen darüberliegenden Stufen bleibt jeweils ein Tabelleneintrag frei.
  Geben Sie nnn zu niedrig an, stellt BALTER sicher, dass mindestens ein Eintrag möglich ist.
- Format 2 der Anweisung wirkt auf einstufige Tabellen die BALTER neu aufbaut, einschließlich Listen.
- Falls Sie FILLING nicht angeben, bleibt auch auf Stufe 0 ein Eintrag frei.
- Sie können beide Formate gleichzeitig für denselben Setnamen angeben. Dabei können ggf. mehr freie Tabelleneinträge entstehen, als mit Format 1 angegeben.
- Sie können die Anweisungen wiederholen und damit vorangegangene Anweisungen desselben Formats ergänzen und korrigieren.
  Es gilt also jeweils für jeden Setnamen die letzte Eingabe.

6.11.5 Kommandofolge zum Starten von BALTER

Die hier beschriebenen Kommandofolgen gehen davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

Sie können die Analysephase und die Umstrukturierungsphase von BALTER unter der Kennung, in der die Datenbank katalogisiert ist, mit folgenden Kommandos starten (Sie können das Programm auch mit dem Aliasnamen BALTER starten):

Analysephase

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
/START-UDS-BALTER
EXECUTION IS NO.
REPORT IS YES.
END.
```

Umstrukturierungsphase

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
[/CREATE-FILE FILE-NAME=arbeitsdatei-1 ...
 /ADD-FILE-LINK LINK-NAME=SCRTCH1,FILE-NAME=arbeitsdatei-1
     ,ACCESS-METHOD=*UPAM]
[/CREATE-FILE FILE-NAME=arbeitsdatei-2 ...
 /ADD-FILE-LINK LINK-NAME=SORTWK,FILE-NAME=arbeitsdatei-2
     ,ACCESS-METHOD=*UPAM]
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
/START-UDS-BALTER
[SORTCORE IS nnn.]
[FILL feldname OF satzname WITH wert.]
[FILLING IS nnn PERCENT
   [ IN SET NAME IS {setname,... | *ALL [ EXCEPT setname,..]}].]
[FILLING WITH POPULATION
   [ IN SET NAME IS {setname,... | *ALL [ EXCEPT setname,..]}].]
EXECUTION IS YES.
REPORT IS {YES | NO}.
END
```

6.11.6 Beschreibung der BALTER-Meldungen

Anhand der Meldungen, die BALTER auf SYSOUT ausgibt, können Sie die Aktivitäten von BALTER verfolgen:

Meldung	Bedeutung
`*** ANALYSE-PHASE ***`	Startmeldung der Analysephase
`*** REPORT-PHASE ***`	Startmeldung der REPORT-Phase mit dem Analyseprotokoll
`*** EXECUTION-PHASE ***`	Startmeldung der Umstrukturierungsphase
`NO ERRORS DETECTED IN SCHEMA CHANGES`	Keine Fehler in den Schemaänderungen gefunden
`ERRORS DETECTED IN SCHEMA CHANGES`	Fehler in den Schemaänderungen gefunden
`DATABASE ALTERED`	Datenbestand an die Schemaänderungen angepasst
`DATABASE NOT ALTERED`	Datenbestand nicht an die Schemaänderungen angepasst
`NUMBER OF DATABASE ACCESSES` *`ganzzahl`*	Anzahl der Zugriffe auf die DB
`NUMBER OF FILE ACCESSES` *`ganzzahl`*	Anzahl der Dateizugriffe
`NUMBER OF SORT ACCESSES` *`ganzzahl`*	Anzahl der Sortierzugriffe

Tabelle 43: allgemeine BALTER-Meldungen

Aktionsindikatoren

Meldung	Bedeutung

`PRINTOUT OF THE USED TAB2-INDICES`
`TAB2-INDEX` *`nummer`* `FOR RECORD` *`satzname`*
`PRINTOUT OF TAB3- & TAB4-INDICES FOR MATCHING AND SINGULAR SETS`
`TAB3-INDEX` *`nummer`* `FOR SET` *`setname`*
`TAB4-INDEX` *`nummer`* `FOR SET` *`setname`* `KEY-REF` *`keyref`*
	Die TAB-INDICES dienen ausschließlich Diagnosezwecken;
	sie geben Aufschluss über die Veränderungen und Aktivitäten von BALTER an Satzarten, Sets und Schlüsseln.

Tabelle 44: TAB-INDICES

Umstrukturierungsmeldungen

Während der Umstrukturierungsphase protokolliert BALTER auf SYSOUT alle Änderungen, die er an Satzarten, Sets oder Schlüsseln vornimmt:

Meldung	Bedeutung
`REALM ADDED TO DATABASE: realmname`	Realm *realmname* zur Datenbank hinzugefügt
`RECORD DELETED FROM DATABASE: satzname`	Satzart *satzname* aus der Datenbank gelöscht
`RECORD ADDED TO DATABASE: satzname`	Satzart *satzname* zur Datenbank hinzugefügt

`RECORD MODIFICATION STARTED FOR:`
`REC NAME: satzname`
`REC REF : satzartnummer`
	Ändern der Satzart *satzname* begonnen

`SET REF : setnummer` `SET NAME: setname`	Ändern/Aufbauen des Sets *setname* begonnen
`CALCKEY TABLE`	Ändern/Aufbauen der CALC-Key-Tabelle begonnen
`SORTKEY TABLE, DBTT COLUMN NR: ganzzahl`	Ändern/Aufbauen der SORT-Key-Tabelle begonnen; DBTT-Spalten-Nr.: *ganzzahl*
`SEARCHKEY TABLE, DBTT COLUMN NR: ganzzahl`	Ändern/Aufbauen der mehrstufigen SEARCH-Key-Tabelle begonnen; DBTT-Spalten-Nr.: *ganzzahl*
`TABLE FILLING IS ganzzahl PERCENT`	Füllgrad für die angegebene Tabelle ist *ganzzahl* Prozent; erscheint nur, wenn FILLING (Format 1) angegeben wurde
`MINIMUM TABLE SIZE FROM POPULATION:` *ganzzahl* `ENTRIES`	Die Tabellengröße wurde auf Grund der POPULATION-Klausel bestimmt, und die Tabelle enthält *ganzzahl* Einträge; erscheint nur, wenn FILLING (Format 2) angegeben wurde
`CALC SEARCHKEY TABLE`	Aufbau des indirekten Hashbereichs für den CALC-SEARCH-Key begonnen
`ALLOCATION OF LIST RECORDS STARTED`	Speichern der Sätze in die Liste begonnen
`STORING DATABASE RECORDS`	Speichern der Sätze in die Datenbank begonnen
`DELETION OF REALM: realmname`	Realm *realmname* aus der Datenbank gelöscht

Tabelle 45: Umstrukturierungsmeldungen

6.12 Zugriffsberechtigungen anpassen

Der Umstrukturierungsprozess ändert nichts an den Zugriffsberechtigungen, die Sie mit dem Dienstprogramm BPRIVACY in die alte Datenbank eingetragen hatten.

Diese Zugriffsberechtigungen müssen Sie an das neue Schema anpassen, d.h. Sie müssen sie neu eintragen.

Falls vor der Umstrukturierung keine Benutzergruppennamen für Zugriffsrechte vergeben waren, können Sie auf diesen Verarbeitungsschritt verzichten.

6.13 Subschemata anpassen

Beim Vorbereiten der Compilerdatenbank auf die Umstrukturierung löscht BCHANGE unter anderem alle SSIA’s im DBDIR und alle Subschema-Informationen im DBCOM; der DDL-Compiler bereitet anschließend beim Übersetzen der neuen Schema-DDL den COSSD für die Aufnahme neuer Subschema-Informationen vor. Nach der Umstrukturierungsphase sind also alle alten Subschema-Informationen gelöscht; im neuen COSSD sind noch keine Subschema-Informationen eingetragen.

Daher müssen Sie, nachdem BALTER Ihre Datenbank umstrukturiert hat, alle Subschemata neu übersetzen, jeweils eine neue SSIA erzeugen und diese im DBDIR eintragen.

6.13.1 Kompatible Subschemata übernehmen

Oft sind nicht alle Subschemata von den Änderungen des Schemas betroffen. Daher kopiert BCHANGE zu Beginn der Umstrukturierung den COSSD in die Datei COSSD.O, sodass trotz Umstrukturierung noch alle alten Subschema-Informationen vorhanden sind. Wollen Sie diese alten Subschemata übernehmen, so müssen Sie, nachdem BALTER die Umstrukturierungsphase erfolgreich abgeschlossen hat, mit dem DDL-Compiler einen Lauf zum Übernehmen der alten Subschemata durchführen.

Bei dem Compilerlauf zum Übernehmen der Subschemata liest der DDL-Compiler aus der Datei COSSD.O alle alten Subschemata aus und prüft sie durch Neuübersetzen auf Verträglichkeit mit dem neuen Schema. Dabei unterscheidet er drei Fälle:

- die alte Subschema-Beschreibung ist mit dem neuen Schema unverträglich
- das alte Subschema ist auf Grund von logischen und/oder physischen Veränderungen im Schema mit dem neuen Schema unverträglich, d.h. die Ausführung von DML-Anweisungen ist betroffen
- das alte Subschema ist von Änderungen im neuen Schema nicht betroffen

In den ersten beiden Fällen legt der DDL-Compiler weder im DBCOM noch im COSSD Subschema-Information ab. Nur im letzten Fall, wenn ein Subschema nicht von den Schemaänderungen betroffen ist, übernimmt er es aus der Datei COSSD.O, compiliert es neu und trägt die Subschema-Information in den neuen DBCOM und in den neuen COSSD ein. Für jedes übernommene Subschema müssen Sie anschließend mit dem Dienstprogramm BGSSIA eine neue SSIA generieren und diese im DBDIR eintragen.

Beachten Sie bitte, dass "Verträglichkeit" nur bedeutet, dass die Sicht des alten Subschemas auf das neue Schema die gleiche geblieben ist wie auf das alte Schema. Es bedeutet nicht, dass etwa bei Verwenden der "COPY [ALL] RECORD[S]"-Klausel automatisch die Sicht auf die (aufwärtskompatiblen) Änderungen im neuen Schema gewährt wird; möchten Sie diese erlangen, müssen Sie das Subschema neu übersetzen.

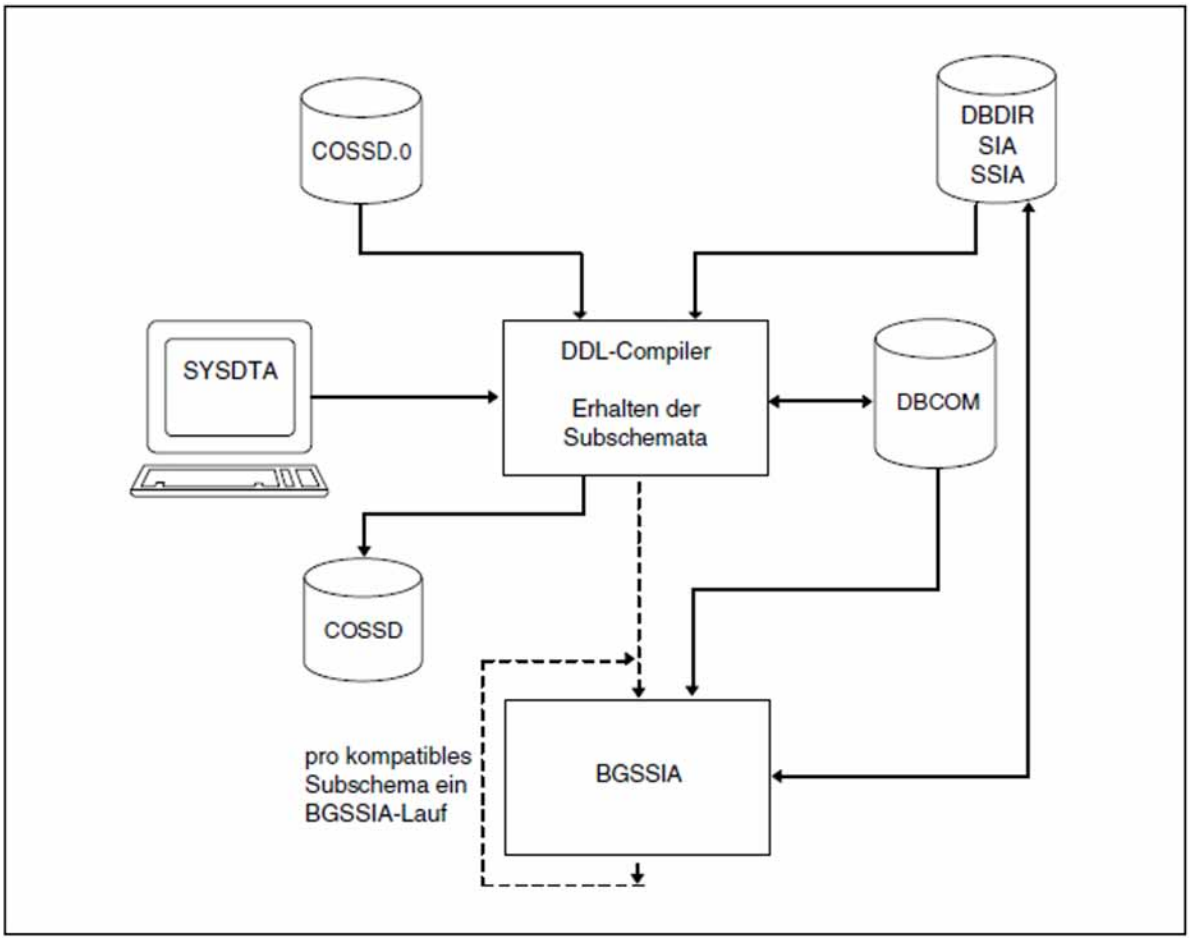

Bild 31: Systemumgebung beim Übernehmen der Subschemata

Den Compilerlauf zum Übernehmen der kompatiblen Subschemata können Sie wahlweise durchführen; lassen Sie ihn weg, so müssen Sie alle Subschemata einzeln neu übersetzen und die jeweilige SSIA neu generieren und in das DBDIR eintragen.

Verträglichkeiten/Unverträglichkeiten der Subschemata

schemaname und PRIVACY LOCK FOR COPY.....

eine Änderung des Schemanamens und der PRIVACY LOCK-Angaben wirkt sich nicht auf das Übernehmen der Subschemata aus!
Diese Änderungen müssen Sie erst bei nachfolgenden Subschema-Übersetzungen berücksichtigen.

PRIVACY LOCK FOR COMPILE

bei dem Compilerlauf zum Übernehmen der Subschemata kopiert der DDL-Compiler diese PRIVACY-Angaben aus der **alten** Subschema-Beschreibung, sodass Zugriffssperren für das Übersetzen von Anwenderprogrammen erhalten bleiben.

bezeichner

nicht verträglich ist ein altes Subschema mit dem neuen Schema, wenn Sie in der LOCATION MODE-Klausel, der WITHIN-Klausel (Satzartebene) oder der SET OCCURRENCE SELECTION-Klausel einen *bezeichner* hinzugefügt oder gelöscht oder den bisherigen *bezeichner* umbenannt haben, sofern die entsprechende Satzart bzw. der Set in dem Subschema liegt.

Anweisungen zum Übernehmen der Subschemata

Der DDL-Compiler benötigt zum Übernehmen der Subschemata folgende Anweisungen:

Anweisung	Standardwert	Bedeutung
COMPARE SUBSCHEMAS	-	Übernehmen der Subschemata veranlassen
[SORCLIST IS { YES | NO }]	YES	Listing der Subschemata ausdrucken
[DIAGNOSTIC { YES | NO }]	NO	Unverträglichkeiten der zum neuen Schema inkompatiblen Subschemata diagnostizieren und in Form von Fehlermeldungen auflisten
END	-	Eingabe der Anweisungen beenden

Tabelle 46: Anweisungen zum Übernehmen der Subschemata

Kommandofolge für das Übernehmen der Subschemata

Mit folgenden Kommandos können Sie einen DDL-Compilerlauf zum Übernehmen der Subschemata starten (siehe Abschnitt „Schema-DDL übersetzen"):

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /START-UDS-DDL
04 COMPARE SUBSCHEMAS
05 [DIAGNOSTIC {YES | NO}]
06 [SORCLIST IS {YES | NO}]
07 END
```

Für jedes übernommene Subschema müssen Sie mit folgenden Kommandos anschließend die SSIA erzeugen und im DBDIR eintragen (siehe Abschnitt „Subschema Information Area (SSIA) erzeugen mit BGSSIA"):

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /START-UDS-BGSSIA
04 GENERATE SUBSCHEMA subschemaname OF SCHEMA schemaname
05 [DISPLAY[ SUBSCHEMA subschemaname OF SCHEMA schemaname]]
06 END
```

6.13.2 Inkompatible Subschemata anpassen

Bei allen Subschemata, die in ihrer ursprünglichen Form mit dem neuen Schema nicht verträglich sind, müssen Sie folgende Maßnahmen durchführen:

- eventuell die Subschema-Beschreibung korrigieren
- mit dem DDL-Compiler das korrigierte Subschema neu übersetzen
- mit BGSSIA eine neue SSIA generieren und im DBDIR eintragen
- alle entsprechenden Benutzerprogramme neu übersetzen und binden

Die hier beschriebenen Kommandofolgen geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme". Dort finden Sie auch die möglichen Aliasnamen für den Aufruf).

Kommandofolge zum Anpassen der Subschemata

Korrigiertes Subschema übersetzen: (siehe Abschnitt „Schema-DDL übersetzen")

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /START-UDS-DDL
04 [DELETE 'subschemaname':'neuer schemaname']
05 SOURCE IS 'subschemadatei'
06 SORCLIST IS YES
07 END
08 /ASSIGN-SYSDTA TO=*SYSCMD
```

02 Versionsabhängiges Modul des linked-in DBH der entsprechenden Version wird nachgeladen (siehe Handbuch „Anwendungen programmieren“, Abschnitt „UDS/SQL-TIAM-Anwenderprogramme binden, laden und starten“).

03 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen DDL gestartet werden.

04 Die Anweisung DELETE dürfen Sie nur angeben, wenn Sie ein Subschema geändert haben und neu übersetzen wollen, das der DDL-Compiler beim Übernehmen der Subschemata als kompatibel erkannt und bereits übernommen hatte.

SSIA erzeugen und in das DBDIR eintragen

Siehe Abschnitt „Subschema Information Area (SSIA) erzeugen mit BGSSIA".

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /START-UDS-BGSSIA
04 GENERATE SUBSCHEMA subschemaname OF SCHEMA schemaname
05 [DISPLAY[ SUBSCHEMA subschemaname OF SCHEMA schemaname]]
06 END
```

6.14 DB-Anwendungen anpassen

Nach dem Anpassen der Subschemata und der Zugriffsberechtigungen an die umstrukturierte Datenbank müssen Sie alle DB-Anwenderprogramme, die ein inkompatibles Subschema verwenden - den neuen Definitionen entsprechend - gegebenenfalls korrigieren, in jedem Fall aber neu übersetzen und binden.

DB-Anwenderprogramme, die ein kompatibles Subschema verwenden, müssen Sie nicht neu übersetzen und binden; das gilt auch für SQL-Anwenderprogramme.

Hinweis für UDS-D

Gegebenenfalls müssen geänderte Subschemamodule zum entfernten Anwenderprogramm (siehe Handbuch „Datenbankbetrieb") transportiert werden.

6.15 Probable Position Pointer (PPP) aktualisieren

Sind sowohl im alten als auch im neuen Schema Adressverweise vorhanden, die als Probable Position Pointer (PPP) angelegt sind, so werden diese beim Verlagern von Daten während der Umstrukturierung nicht in allen Fällen aktualisiert.

Beim vollständigen oder teilweisen Verlagern von Sätzen:

- Adressverweise in Tabellen oder indirekten Hashbereichen zu den Sätzen aktualisiert BALTER nur, wenn die Tabellen oder indirekten Hashbereiche auf Grund der Schemaänderungen neu aufgebaut werden müssen.
- Adressverweise innerhalb der Sätze einer Kette aktualisiert BALTER nicht.
- Adressverweise in Membersätzen zu den Ownersätzen aktualisiert BALTER immer, wenn die Ownersätze verlagert werden.

  Ein Aktualisieren der Adressverweise zu Sätzen ist mit dem Dienstprogramm BREORG möglich. Mit der Anweisung REORGANIZE-POINTERS können Sie alle in einem Realm enthaltenen Probable Position Pointer (PPP) pauschal anpassen.

Beim Neuaufbau, Löschen, Verlagern von Tabellen:

- Adressverweise in Ownersätzen zu ihren Tabellen behandelt BALTER als Act-Keys.

  Die Adressverweise werden in allen betroffenen Fällen als Act-Keys realisiert:
  - wenn die Tabelle neu aufgebaut wird
  - wenn vorhandene leere Tabellen gelöscht werden
  - wenn Tabellen in einen anderen Realm verlagert werden
  - wenn leere einstufige Listen in einen anderen Realm verlagert werden
  - wenn die Adressverweise neu hinzugefügt werden

Sind nach der Umstrukturierung Probable Position Pointer (PPP) mit veralteten Werten vorhanden, können bei DB-Anwendungen Laufzeitveränderungen auftreten.

6.16 Maßnahmen vor Wiederaufnahme des DB-Betriebs

Falls vor dem Umstrukturierungszyklus das After-Image-Logging ausgeschaltet wurde, ist dadurch eine Logging-Lücke entstanden. Nach dem Umstrukturierungszyklus kann nun mit dem Dienstprogramm BMEND das After-Image-Logging wieder eingeschaltet werden (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BMEND). Danach muss erneut eine Sicherung der Datenbank erzeugt werden (siehe Handbuch „Datenbankbetrieb“, Datenbank sichern und wiederherstellen im Fehlerfall).

Die Dateien DBCOM.O und COSSD.O sowie Benutzerrealms, die im neuen Schema nicht mehr vorkommen, können Sie nun wieder löschen.

6.17 Beispiel

Die im folgenden Diagramm skizzierte Datenbank VERS soll folgendermaßen umstrukturiert werden:

- der Realm TRANSPORTRLM wird hinzugefügt
- die Satzart TRANSPORTVERSICHERUNG wird in den Realm TRANSPORTRLM verlegt
- der Set ABGESCHL-TRANSPORT mit der Ownersatzart KUNDE und der Membersatzart TRANSPORTVERSICHERUNG wird hinzugefügt
- der Set ANSPRUECHE-TRANSPORT mit der Ownersatzart TRANSPORTVERSICHERUNG und der Membersatzart SCHADENSMELDUNG wird hinzugefügt
- die Satzart KUNDE wird modifiziert

Das folgende Diagramm skizziert das Schema der Datenbank VERS nach dem Umstrukturieren (VERS vor dem Umstrukturieren siehe Bild 3, "Beispieldatenbanken").

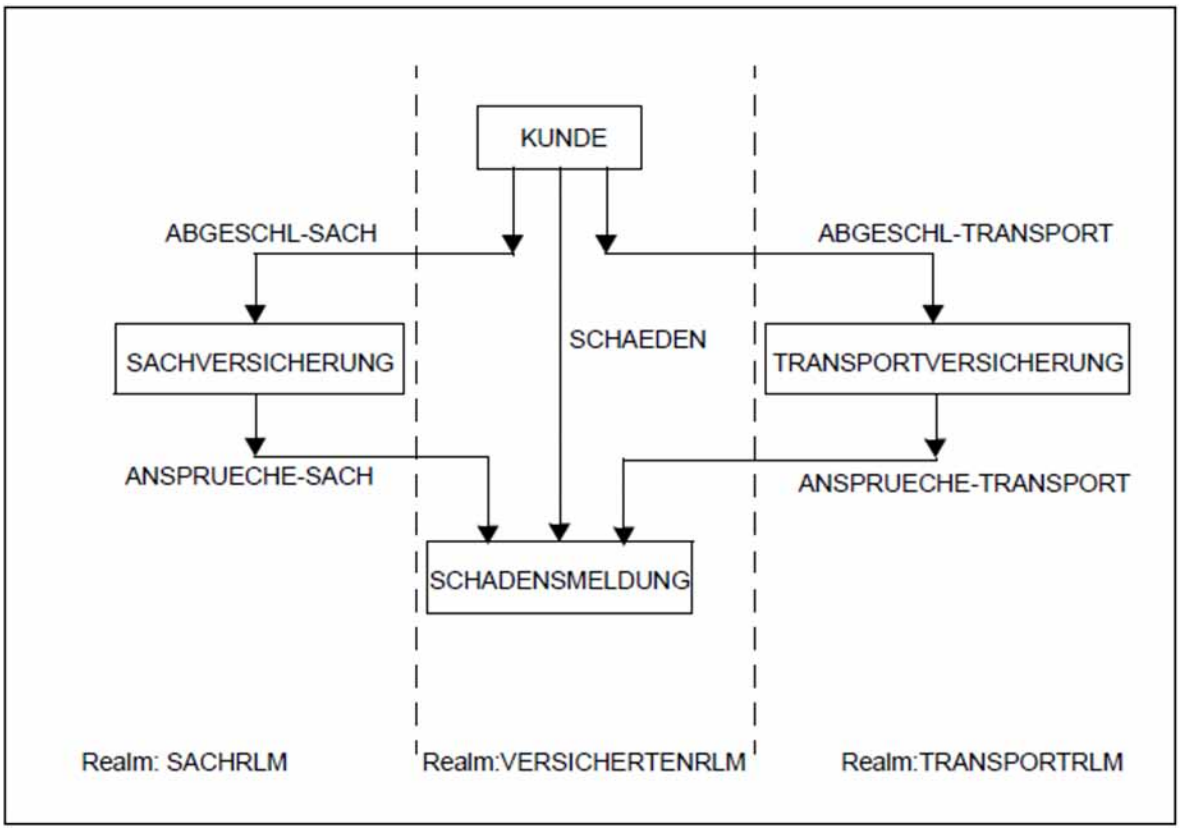

Bild 32: Datenbank VERS nach dem Umstrukturieren

Vor dem Umstrukturieren werden DBDIR, DBCOM, COSSD, HASHLIB und der bei der Umstrukturierung benötigte Benutzerrealm gesichert (VORUM). Nach dem Umstrukturieren wird die ganze Datenbank gesichert (NACHUM).

Jeweils vor dem Erstellen der Sicherung sollten Sie mit dem Dienstprogramm BCHECK die Konsistenz der Datenbank prüfen (siehe Handbuch „Sichern, Informieren und Reorganisieren“).

Dieses Beispiel soll nur den Ablauf einer Umstrukturierung verdeutlichen; daher wurde das Schema einfach gewählt und die Protokolle zur Schema-DDL, SSL usw. weggelassen.

DBDIR, DBCOM, COSSD und HASHLIB sichern

```
/COPY-FILE FROM-FILE=VERS.DBDIR,TO-FILE=VERS.DBDIR.VORUM
/COPY-FILE FROM-FILE=VERS.DBCOM,TO-FILE=VERS.DBCOM.VORUM
/COPY-FILE FROM-FILE=VERS.COSSD,TO-FILE=VERS.COSSD.VORUM
/COPY-FILE FROM-FILE=VERS.HASHLIB,TO-FILE=VERS.HASHLIB.VORUM
```

BCHANGE-Lauf und neue Schema-DDL und SSL übersetzen

Die Schema-DDL ist bei diesem Lauf noch fehlerhaft. Dabei handelt es sich um einen Fehler, der erst bei der Übersetzung der SSL entdeckt wird.

```
/START-UDS-BCHANGE
***** START   BCHANGE   (UDS/SQL V2.9 0000 )  2017-06-28  11:33:04

***** THE FILE: :SQL2:$XXXXXXXX.VERS.DBCOM IS COPIED TO:
     :SQL2:$XXXXXXXX.VERS.DBCOM.O

***** THE FILE: :SQL2:$XXXXXXXX.VERS.COSSD IS COPIED TO:
     :SQL2:$XXXXXXXX.VERS.COSSD.O
1006 RESTRUCTURING SUCCESSFULLY INITIATED

***** DIAGNOSTIC SUMMARY OF BCHANGE

              NO WARNINGS
              NO ERRORS
              NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          94
***** NORMAL END   BCHANGE     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:05

/CREATE-FILE FILE-NAME=VERS.DBSTAT
/CREATE-FILE FILE-NAME=VERS.DBSTAT.SAVE

/START-UDS-DDL
***** START       DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:05
*  DDLCOMP: INPUT SYSTEMPARAMETERS
SOURCE IS 'S.VERS.DDL.NEU'
END
*  DDLCOMP: READ SCHEMA/SUBSCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:05/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:05/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  DDLCOMP: START SCHEMA-PHASE
*  DDLCOMP: CHECK SCHEMA RULES
*  DDLCOMP: CHECK DATA ALLOCATION
*  DDLCOMP: SEMANTIC TEST
*  DDLCOMP: CYCLUS TESTS
*  DDLCOMP: ERROR DIAGNOSTIC
*  DDLCOMP: NO ERRORS IN SCHEMA-PHASE
*  DDLCOMP: CREATE FILE COSSD
*  DDLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:05/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
```

```
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                     651      1999        67         914         39
%  UDS0213 UDS NORMAL BEENDET MIT *************651 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:05/0YA2)
***** DIAGNOSTIC SUMMARY FOR DDL-SCHEMA KUNDENDATEI

              NO ERRORS
+++++          9 WARNINGS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:05

/START-UDS-SSL
***** START       SSLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:05
*  SSLCOMP: INPUT SYSTEMPARAMETERS
SOURCE IS 'S.VERS.SSL.NEU'
END
*  SSLCOMP: READ SSL-SCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:05/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:05/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  SSLCOMP: START SSL-PHASE
*  SSLCOMP: CHECK SSL RULES
*  SSLCOMP: SEMANTIC TEST
*  SSLCOMP: ERROR DIAGNOSTIC
*  SSLCOMP: ERRORS DETECTED IN SSL-PHASE
*  SSLCOMP: ERRORS DETECTED
*  SSLCOMP: ALL SSL-OPTIONS ARE RESET
+++++ ERROR: 0012 UDS-DBH RETURNS WITH DATABASE-STATUS '04021'
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:05/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                     303        387         61         71         25
%  UDS0213 UDS NORMAL BEENDET MIT *************303 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:05/0YA2)

***** DIAGNOSTIC SUMMARY FOR SSL - SCHEMA

+++++          2 ERRORS
              NO WARNINGS

***** END OF DIAGNOSTIC SUMMARY
+++++ ABNORMAL END SSLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:05
```

Korrigiertes Schema übersetzen

Nachdem die Schema-DDL entsprechend der SSL-ERROR-DIAGNOSTIC korrigiert wurde, muss das bereits eingetragene, aber fehlerhafte Schema gelöscht werden. Erst danach kann die korrigierte Schema-DDL und anschließend die SSL übersetzt werden.

```
/START-UDS-DDL
***** START       DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:06
*  DDLCOMP: INPUT SYSTEMPARAMETERS
DELETE SCHEMA 'KUNDENDATEI'
END
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:06/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:06/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  DDLCOMP: SCHEMA HAS BEEN ERASED
*  DDLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:06/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                      6       1075         70        556         39
%  UDS0213 UDS NORMAL BEENDET MIT ***************6 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:06/0YA2)

/START-UDS-DDL
***** START       DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:06
*  DDLCOMP: INPUT SYSTEMPARAMETERS
SOURCE IS 'S.VERS.DDL.KORR'
DISPLAY IS YES
END
*  DDLCOMP: READ SCHEMA/SUBSCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:06/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:06/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  DDLCOMP: START SCHEMA-PHASE
*  DDLCOMP: CHECK SCHEMA RULES
*  DDLCOMP: CHECK DATA ALLOCATION
*  DDLCOMP: SEMANTIC TEST
*  DDLCOMP: CYCLUS TESTS
*  DDLCOMP: ERROR DIAGNOSTIC
*  DDLCOMP: NO ERRORS IN SCHEMA-PHASE
*  DDLCOMP: DISPLAY SCHEMA
*  DDLCOMP: CREATE FILE COSSD
*  DDLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:06/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                    751       2120         66        914         40
%  UDS0213 UDS NORMAL BEENDET MIT *************751 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:06/0YA2)

***** DIAGNOSTIC SUMMARY FOR DDL-SCHEMA KUNDENDATEI

              NO ERRORS
+++++          9 WARNINGS
```

```
***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:06


/START-UDS-SSL
***** START       SSLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:06
*  SSLCOMP: INPUT SYSTEMPARAMETERS
SOURCE IS 'S.VERS.SSL.NEU'
END
*  SSLCOMP: READ SSL-SCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:06/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:06/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  SSLCOMP: START SSL-PHASE
*  SSLCOMP: CHECK SSL RULES
*  SSLCOMP: SEMANTIC TEST
*  SSLCOMP: ERROR DIAGNOSTIC
*  SSLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:06/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                    127        253         63         34         23
%  UDS0213 UDS NORMAL BEENDET MIT *************127 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:06/0YA2)

***** DIAGNOSTIC SUMMARY FOR SSL - SCHEMA

             NO ERRORS
             NO WARNINGS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   SSLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:06
/DELETE-SYSTEM-FILE FILE-NAME=*OMF

/START-UDS-BGSIA
***** START       BGSIA       (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:06
GENERATE SCHEMA KUNDENDATEI
DISPLAY
END
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:06/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:06/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
ESTIMATE-REPORT
********** FOR USER-REALM     3 NAME IS : SACHRLM
      A SIZE OF         24 BLOCKS WAS ESTIMATED

********** FOR USER-REALM     4 NAME IS : VERSICHERTENRLM
      A SIZE OF        239 BLOCKS WAS ESTIMATED

********** FOR USER-REALM     6 NAME IS : TRANSPORTRLM
      A SIZE OF         24 BLOCKS WAS ESTIMATED
END OF ESTIMATE-REPORT
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:06/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
```

```
 0YA2: VERS                    569        779         60        183         30
%  UDS0213 UDS NORMAL BEENDET MIT *************569 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:06/0YA2)

***** DIAGNOSTIC SUMMARY OF BGSIA

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   BGSIA       (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:06

/MODIFY-JOB-SWITCHES ON=(1,4)
/START-LMS
//MODIFY-LOGGING-PARAMETERS LOG=*MAX
//OPEN-LIBRARY LIB=VERS.HASHLIB,MODE=*UPDATE
//ADD-ELEMENT FROM-FILE=*OMF,TO-ELEM=*LIB-ELEM(TYPE=R),WRITE-MODE=*ANY
INPUT  OMF
OUTPUT LIBRARY= :SQL2:$XXXXXXXX.VERS.HASHLIB
         ADD UDSHASH AS (R)UDSHASH/@(0002)/2017-06-28 , OUTPUT REPLACED
//SHOW-ELEM-ATTR

INPUT  LIBRARY= :SQL2:$XXXXXXXX.VERS.HASHLIB
TYP NAME    VER (VAR#) DATE        NAME    VER (VAR#) DATE
(R) ADMIN## @   (0001) 2017-06-28   UDSHASH @   (0002) 2017-06-28
       2 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS

//END
/MODIFY-JOB-SWITCHES OFF=(1)
```

Analysephase mit REPORT IS YES und EXECUTION IS NO

```
/START-UDS-BALTER
***** START       BALTER      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:07
REPORT    IS YES.
EXECUTION IS NO.
END.

***  ANALYSE-PHASE    ***
*** DATE AND TIME 2017-06-28  11:33:07

+++++ WARNING: 1081 AREAS DELETED FROM RECORD-WITHIN-CLAUSE
RECORD: TRANSPORTVERSICHERUNG
      IF RECORD OCCURRENCES ARE PRESENT IN AREAS
      WHICH ARE DELETED FROM RECORD-WITHIN-CLAUSE
      THE RESTRUCTURING PROCESS WILL END  ABNORMALLY.

NO ERRORS DETECTED IN SCHEMA CHANGES

***  REPORT-PHASE     ***
*** DATE AND TIME 2017-06-28  11:33:07

REALM NOT NEEDED: SACHRLM
REALM NEEDED: VERSICHERTENRLM

DATABASE NOT ALTERED
```

```
NUMBER OF FILE ACCESSES:           0


***** DIAGNOSTIC SUMMARY OF BALTER

+++++           1 WARNINGS
               NO ERRORS
               NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :         107
***** NORMAL END   BALTER      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:07
```

Da kein Satz der Satzart TRANSPORTVERSICHERUNG in der Datenbank enthalten ist, kann die Warnung ignoriert und die Datenbank umstrukturiert werden.

Von den beiden Benutzerrealms VERS.SACHRLM und VERS.VERSICHERTENRLM wird nur VERS.VERSICHERTENRLM für die Umstrukturierung benötigt.

Dieser Realm wird gesichert:

```
/COPY-FILE FROM-FILE=VERS.VERSICHERTENRLM, TO-FILE=VERS.VERSICHERTENRLM.VORUM
```

Umstrukturierungsphase

```
/CREATE-FILE FILE-NAME=VERS.TRANSPORTRLM,SUPPORT=*PUBLIC-DISK( -
/   PRIMARY-ALLOCATION=50,SECONDARY-ALLOCATION=0)
/START-UDS-BALTER
***** START       BALTER      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:07
REPORT    IS NO .
EXECUTION IS YES.
END.

***  ANALYSE-PHASE    ***
*** DATE AND TIME 2017-06-28  11:33:07

+++++ WARNING: 1081 AREAS DELETED FROM RECORD-WITHIN-CLAUSE
RECORD: TRANSPORTVERSICHERUNG
      IF RECORD OCCURRENCES ARE PRESENT IN AREAS
      WHICH ARE DELETED FROM RECORD-WITHIN-CLAUSE
      THE RESTRUCTURING PROCESS WILL END  ABNORMALLY.

NO ERRORS DETECTED IN SCHEMA CHANGES


***  EXECUTION-PHASE  ***
*** DATE AND TIME 2017-06-28  11:33:07

REALM ADDED TO DATABASE: TRANSPORTRLM
*** DATE AND TIME 2017-06-28  11:33:07

MODIFICATION  CONCERNING OWNER ATTRIBUTE  STARTED FOR
REC NAME: TRANSPORTVERSICHERUNG
REC REF:     3
*** DATE AND TIME 2017-06-28  11:33:07

MODIFICATION  CONCERNING OWNER ATTRIBUTE  STARTED FOR
REC NAME: KUNDE
REC REF:     4
*** DATE AND TIME 2017-06-28  11:33:07

RECORD MODIFICATION STARTED FOR:
REC NAME: TRANSPORTVERSICHERUNG
REC REF:     3
*** DATE AND TIME 2017-06-28  11:33:07

RECORD MODIFICATION STARTED FOR:
REC NAME: KUNDE
REC REF:      4
*** DATE AND TIME 2017-06-28  11:33:07
0074 REALM VERSICHERTENRLM HAS BEEN EXTENDED BY       64 DATABASE-PAGES
      NEW NR OF PAGES           :             190
```

```
CALCKEY TABLE
*** DATE AND TIME 2017-06-28  11:33:07

STORING DATABASE RECORDS
*** DATE AND TIME 2017-06-28  11:33:07


RECORD MODIFICATION STARTED FOR:
REC NAME: SCHADENSMELDUNG
REC REF:      5
*** DATE AND TIME 2017-06-28  11:33:07

DATABASE ALTERED
*** DATE AND TIME 2017-06-28  11:33:07

NUMBER OF FILE ACCESSES:           10

***** DIAGNOSTIC SUMMARY OF BALTER

+++++           1 WARNINGS
               NO ERRORS
               NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          248
***** NORMAL END   BALTER       (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:07
```

Zugriffsberechtigungen neu eintragen

```
/START-UDS-BPRIVACY
***** START       BPRIVACY     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:07
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:07/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:07/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
%  UDS0722 UDS AUFTRAG ADD RLOG 150628093306 WIRD AUSGEFUEHRT (ILL1283,11:33:07/0YA2)
%  UDS0356 UDS DURCHFUEHRUNG DER AUFTRAEGE FUER VERS     TERMINATED (ILL1309,11:33:07/0YA2)
//ADD-USER-GROUP USER-GROUP-NAME=*FREE-FORMAT(HOST=D017ZE07,USER-ID=XXXXXXXX), -
// OBJECT=(*REALM(NAME=*ALL,RIGHT=ALL),*RECORD(NAME=*ALL,RIGHT=ALL),*SET(NAME=*ALL,
RIGHT=ALL))
//END
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:07/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                    11         115         57         36         23
%  UDS0213 UDS NORMAL BEENDET MIT **************11 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:07/0YA2)

***** DIAGNOSTIC SUMMARY OF BPRIVACY

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   BPRIVACY     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:07
```

Test, ob das Subschema zum neuen Schema kompatibel ist

```
/START-UDS-DDL
***** START       DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:07
*  DDLCOMP: INPUT SYSTEMPARAMETERS
COMPARE SUBSCHEMAS
DIAGNOSTIC IS YES
END
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:07/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:07/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------

*  DDLCOMP: READ SCHEMA/SUBSCHEMA     1
*  DDLCOMP: START SUBSCHEMA-PHASE
*  DDLCOMP: CHECK SUBSCHEMA RULES
*  DDLCOMP: CHECK DATA ALLOCATION
*  DDLCOMP: SUBCOPY
*  DDLCOMP: ERROR DIAGNOSTIC
*  DDLCOMP: ERRORS DETECTED IN SUBSCHEMA-PHASE
*  DDLCOMP: SUBSCHEMA HAS BEEN ERASED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:07/0YA2)
 0YA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                     832       2616         74        936         44
%  UDS0213 UDS NORMAL BEENDET MIT *************832 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:07/0YA2)

***** DIAGNOSTIC SUMMARY FOR DDL-SUBSCHEMA

+++++           1 ERRORS
+++++           9 WARNINGS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   DDLCOMP      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:07
```

Subschema anpassen

Da die alte Subschema-DDL zur neuen Schema-DDL nicht passt, wird die Subschema-DDL korrigiert und anschließend neu übersetzt.

```
/START-UDS-DDL
***** START       DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:07
*  DDLCOMP: INPUT SYSTEMPARAMETERS
SOURCE IS 'S.VERS.SUBDDL.NEU'
END
*  DDLCOMP: READ SCHEMA/SUBSCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:08/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:08/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  DDLCOMP: START SUBSCHEMA-PHASE
*  DDLCOMP: CHECK SUBSCHEMA RULES
*  DDLCOMP: CHECK DATA ALLOCATION
*  DDLCOMP: SUBCOPY
*  DDLCOMP: ERROR DIAGNOSTIC
```

```
*  DDLCOMP: NO ERRORS IN SUBSCHEMA-PHASE
*  DDLCOMP: WRITE SUBSCHEMA ON COSSD
*  DDLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:08/0YA2)
 0YA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                    1363       2581         76        631         49
%  UDS0213 UDS NORMAL BEENDET MIT ************1363 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:08/0YA2)

***** DIAGNOSTIC SUMMARY FOR DDL-SUBSCHEMA

             NO ERRORS
             NO WARNINGS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:08
/START-UDS-BGSSIA
***** START        BGSSIA      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:08
GENERATE SUBSCHEMA VERWALTUNG OF SCHEMA KUNDENDATEI
DISPLAY
END
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:08/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:08/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*** SSIA GENERATION NORMALLY ENDED.
*GENERATION OF ITEM-TABLE AND NAME-TABLE STARTED.
*GENERATION OF ITEM-TABLE AND NAME-TABLE FINISHED.
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:08/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                    781       1359         76        286         29
%  UDS0213 UDS NORMAL BEENDET MIT *************781 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:08/0YA2)

***** DIAGNOSTIC SUMMARY OF BGSSIA

               NO WARNINGS
               NO ERRORS
               NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   BGSSIA       (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:08
```

Die Umstrukturierung der Datenbank ist hiermit fertig.

Nun müssen selbstverständlich die DB-Anwenderprogramme angepasst werden, so weit sie auf die geänderten Größen Bezug nehmen, und wegen der Inkompatibilität des alten und neuen Subschemas neu übersetzt und gebunden werden.

Umstrukturierte Datenbank reorganisieren

Um Speicherplatz zu sparen, wird der Hashbereich der Satzart KUNDE reorganisiert, d.h. in den vorne im Realm frei gewordenen Bereich zurückverlagert. Der VERSICHERTENRLM kann dadurch wieder verkleinert werden.

```
/START-UDS-BREORG
***** START       BREORG      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:08
//SPECIFY-SCHEMA SCHEMA-NAME=KUNDENDATEI
//REORGANIZE-CALC RECORD-NAME=KUNDE, -
//  CALC-RECORD=*WITHIN-POPULATION(REALM=VERSICHERTENRLM,POPULATION=500),CALC-SEARCHKEY=NONE
//END
***** BEGIN OF CALC-REORGANIZATION     AT 11:33:09
***** RESULTS OF CALC-REORGANIZATION OF RECORD KUNDE
      NEW CALC BEGIN            :           4-         5
      NEW NR OF PRIMARY BUCKETS :          59
      NEW NR OF OVERFLOW BUCKETS:           0
***** END   OF CALC-REORGANIZATION     AT 11:33:09

***** DIAGNOSTIC SUMMARY OF BREORG

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          77
***** NORMAL END   BREORG      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:09

/START-UDS-BREORG
***** START        BREORG      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:09
//SPECIFY-SCHEMA SCHEMA-NAME=KUNDENDATEI
//MODIFY-REALM-SIZE REALM-NAME=VERSICHERTENRLM,REALM-SIZE=MINIMUM
//END
***** BEGIN OF REALM-SIZE-MODIFICATION     AT 11:33:10
***** RESULTS OF FPA-REORGANIZATION OF AREA VERSICHERTENRLM
      NEW FPA FIRST PAGE        : NOT CHANGED
      NEW FPA LAST  PAGE        : NOT CHANGED
      NEW FPA SIZE              : NOT CHANGED
      NEW NR OF PAGES           :              80
***** END   OF REALM-SIZE-MODIFICATION     AT 11:33:10

***** DIAGNOSTIC SUMMARY OF BREORG

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :           70
***** NORMAL END   BREORG       (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:10
```

Maßnahmen vor Wiederaufnahme des Datenbankbetriebs

Es wird eine Schattendatenbank mit dem Suffix NACHUM erstellt. Die Sicherung vor der Umstrukturierung wird gelöscht, ebenso die Dateien VERS.DBCOM.O und VERS.COSSD.O.

```
/COPY-FILE FROM-FILE=VERS.HASHLIB,TO-FILE=VERS.HASHLIB.NACHUM
/COPY-FILE FROM-FILE=VERS.COSSD  ,TO-FILE=VERS.COSSD.NACHUM
/COPY-FILE FROM-FILE=VERS.DBDIR  ,TO-FILE=VERS.DBDIR.NACHUM
/COPY-FILE FROM-FILE=VERS.DBCOM  ,TO-FILE=VERS.DBCOM.NACHUM
/COPY-FILE FROM-FILE=VERS.SACHRLM,TO-FILE=VERS.SACHRLM.NACHUM
/COPY-FILE FROM-FILE=VERS.VERSICHERTENRLM,TO-FILE=VERS.VERSICHERTENRLM.NACHUM
/COPY-FILE FROM-FILE=VERS.TRANSPORTRLM,TO-FILE=VERS.TRANSPORTRLM.NACHUM

/DELETE-FILE FILE-NAME=VERS*VORUM*

/DELETE-FILE FILE-NAME=VERS.DBCOM.O
/DELETE-FILE FILE-NAME=VERS.COSSD.O
```

7 Datenbankobjekte umbenennen (BRENAME, BALTER)

Der Umbenennungszyklus von BRENAME/BALTER ermöglicht, Datenbankobjekte in bestehenden Datenbanken umzubenennen. Dazu werden ausschließlich die Strukturinformationen der Datenbankdateien DBDIR, DBCOM und COSSD bearbeitet. Bei einer Änderung von Namen von Benutzerrealms werden zusätzlich auch einige Strukturinformationen in den entsprechenden Realms geändert.
Die eigentlichen Benutzerdaten (Sätze und Tabellen in den Benutzerrealms) werden im Umbenennungszyklus jedoch weder geprüft noch geändert. Deshalb sind im Umbenennungszyklus auch nur Änderungen zugelassen, die die physische Datenbankstruktur unverändert lassen.

> ¡ Da nur Strukturinformationen bearbeitet werden, kann ein Umbenennungszyklus sehr schnell ablaufen.

Die Aktivitäten, die beim Umbenennen erforderlich sind, gliedern sich in drei Abschnitte:

- vorbereitende Maßnahmen
- Umbenennungsprozess
- Folgeaktivitäten

Vorbereitende Maßnahmen

Beim Umbenennungszyklus kann im Gegensatz zur Umstrukturierung das After Image Logging eingeschaltet bleiben. Nur falls der Name eines Realms geändert werden soll, müssen Sie vor dem Umbenennungszyklus das After-Image-Logging mit dem Dienstprogramm BMEND ausschalten (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BMEND).

Umbenennungsprozess

Dies ist ein Ablauf, der dem Aufbauen einer Datenbank gleicht:

- mit BRENAME bereiten Sie den DBDIR für die Aufnahme einer neuen SIA vor
- anschließend übersetzen Sie Ihre neuen DDL- und SSL-Definitionen und tragen die neue SIA im DBDIR ein
- BALTER prüft die Umbenennung und aktualisiert die Strukturinformationen

Folgende Maßnahmen können bei einem Umbenennungszyklus von BRENAME/BALTER durchgeführt werden:

- Änderung von Feldnamen in Satzarten
- Änderung des Typs von Feldern in Satzarten
- Unterteilung eines bestehenden Feldes in mehrere hintereinanderliegende Einzelfelder
- Umwandlung eines bestehenden Feldes in einen Vektor
- Umwandlung eines oder mehrerer aufeinander folgender Felder in eine Wiederholungsgruppe
- Zusammenfassung mehrerer hintereinanderliegender Einzelfelder zu einem neuen Feld
- Umwandlung eines Vektors zu einem neuen Einzelfeld
- Umwandlung einer Wiederholungsgruppe zu einem oder mehreren aufeinander folgenden Einzelfeldern
- Änderung von Satznamen
- Änderung von Setnamen
- Änderung von Realmnamen

i
- Im Umbenennungszyklus zwischen BRENAME und BALTER kann BMEND nicht ausgeführt werden.
- Der Umbenennungszyklus von BRENAME/BALTER kann nicht mit demUmstrukturierungszyklus von BCHANGE/BALTER kombiniert werden.

Namen von SEARCH-Keys in der DDL/SSL-Source des Schemas bewirken, dass die Deklarationen der SEARCH-Keys von DDL- und SSL-Compiler eindeutig zugeordnet werden können. Wie bei Umstrukturierungen (BCHANGE /BALTER-Zyklus) können diese Namen geändert werden, ohne dass sich damit die Schema- bzw. Subschemabeschreibung in SIA bzw. SSIA ändert. Insofern werden solche Änderungen im Umbenennungszyklus (BRENAME/BALTER-Zyklus) weder in der Analyse noch in den REPORT-Ausgaben des BALTER berücksichtigt.

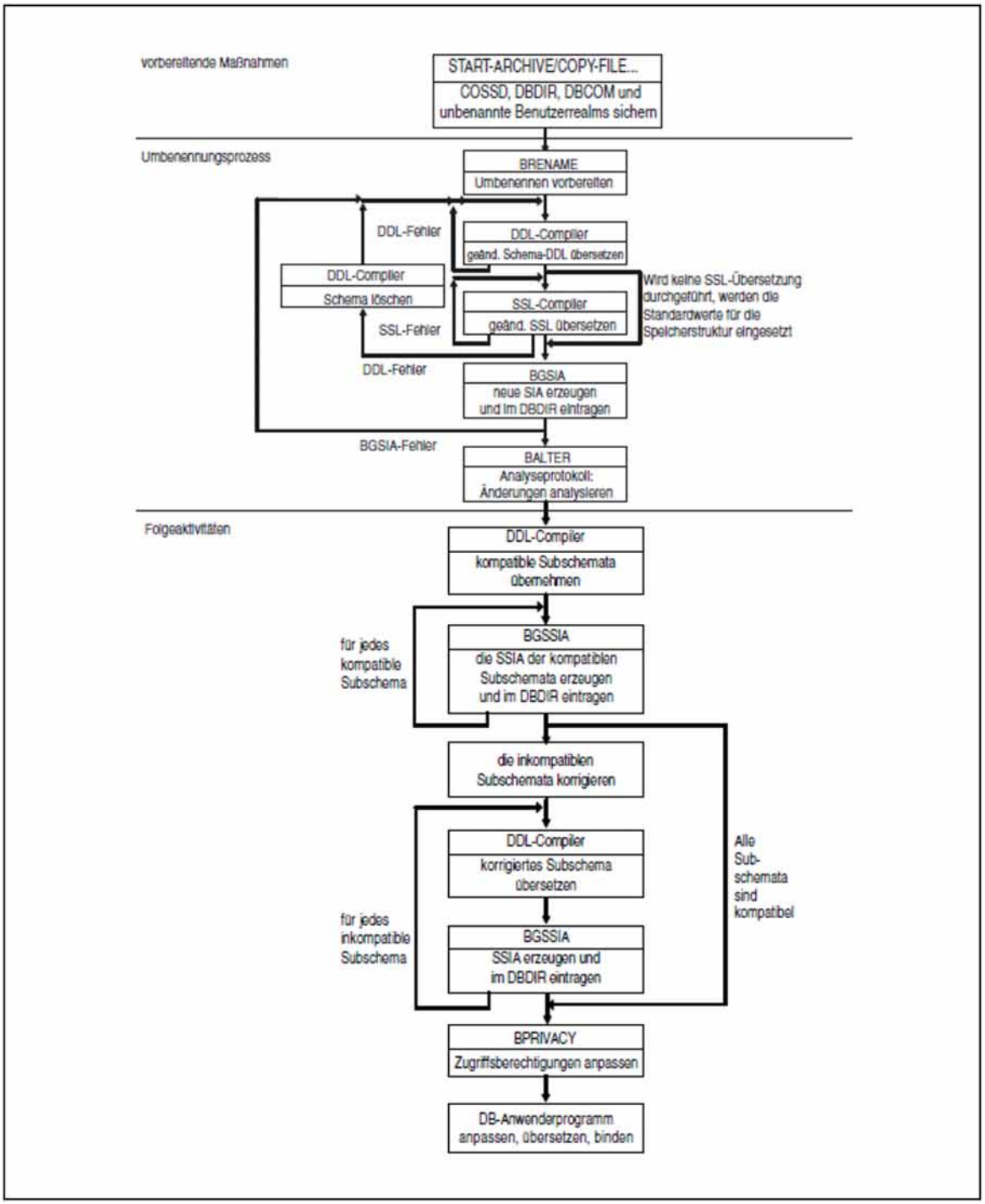

Bild 33: Ablauf einer Umbenennung

Folgeaktivitäten

Nach dem Umbenennen müssen folgende Aktivitäten durchgeführt werden:

- Subschemata an das geänderte Schema anpassen
- DB-Anwenderprogramme an das neue Schema anpassen
- ggf. mit dem Dienstprogramm BCALLSI für CALL-DML-Programme neue SSITAB-Module erzeugen
- ggf. Zugriffsberechtigungen aktualisieren
- ggf. Anwenderprogramme anpassen, übersetzen und binden
- ggf. geänderte Datenbankdateien kopieren und mit BMEND das After-Image-Logging wieder einschalten.

> ¡ Durch den Umbenennungszyklus entsteht eine Logging-Lücke (siehe Handbuch „Datenbankbetrieb“, Langzeitsicherung). Deshalb müssen Sie nach dem Umbenennungszyklus durch Kopieren der geänderten Dateien zusammen mit den nicht geänderten Datenbankdateien eine neue Basis für die Langzeitsicherung aufbauen. Anschließend müssen Sie mit BMEND das After-Image-Logging wieder einschalten.

Datenbestände ändern

Die Benutzerdaten werden in einem Umbenennungszyklus nicht geändert. Sind mit einer Umbenennung semantische Änderungen der Benutzerdaten verbunden, müssen Sie die dazu notwendigen Änderungen der Datenbestände z.B. mit speziellen Anwendungsprogrammen durchführen. Dies kann im normalen Datenbankbetrieb vor oder nach dem Umbenennungszyklus erfolgen (siehe Abschnitt „Benutzerdaten anpassen").

7.1 Schema-DDL ändern

Wollen Sie Ihre Schema-DDL ändern, so müssen Sie eine neue bzw. geänderte, vollständige Schema-Definition erstellen und diese neu übersetzen. Dabei sind jeweils an allen Stellen in der DDL- und SSL-Source die neuen Namen zu nutzen.
BALTER prüft die Umbenennung und aktualisiert die Strukturinformationen.

Damit BGSIA umbenannte Realms, Satzarten und Sets erkennt und die vorhandenen Referenzen unverändert lässt, müssen vorzunehmende Namensänderungen für Realms, Satzarten und Sets vollständig in zusätzlichen Anweisungen übergeben werden.

In der Schema-DDL sind folgende Änderungen möglich:

- Feldnamen in Satzarten ändern
  Die Namen müssen an allen Stellen (z.B. auch bei Schlüsseldefinitionen) geändert werden.
- Typen von Feldern ändern
  Die Länge der Datenfelder in Bytes darf nicht geändert werden. Es sind Typveränderungen von CHAR und nach CHAR möglich. Bei der Umwandlung nach NCHAR werden in dem neuen Feld nur die Hälfte der Zeichen abgespeichert. Typveränderungen von variablen Feldern und von Feldern in komprimierten Sätzen sind nicht erlaubt. Die Benutzerdaten in der Datenbank bleiben auch bei Typänderung physisch unverändert. Es wird nicht geprüft, ob sie mit dem neuen Typ verträglich sind. Typänderungen erfordern daher gesonderte Maßnahmen zur Anpassung der in den Benutzerrealms abgespeicherten Benutzerdaten (siehe „Benutzerdaten anpassen").
  Bei einer Änderung in einen numerischen Typ, sind gegebenenfalls notwendige Ausrichtungen des numerischen Typs zu berücksichtigen.
- ein vorhandenes Feld vom Typ CHAR in mehrere Felder aufteilen
  Die Gesamtlänge muss dabei erhalten bleiben. Wird das Ursprungsfeld als Schlüsselfeld verwendet, muss es als solches durch die Nutzung der neuen Felder erhalten bleiben. Die Besonderheiten bei Änderungen von NCHAR und numerischen Typen gelten entsprechend den Änderungen des Typs eines Einzelfeldes. Die Aufteilung eines Feldes in mehrere erfordert gesonderte Maßnahmen zur Anpassung der in den Benutzerrealms abgespeicherten Benutzerdaten (siehe „Benutzerdaten anpassen").
- vorhandene hintereinanderliegende Felder zu einem Feld vom Typ CHAR zusammenfassen
  Die Gesamtlänge muss dabei erhalten bleiben. Die Zusammenfassung ist nur möglich, wenn nicht einzelne Felder isoliert als Schlüsselfelder genutzt werden. Die Zusammenfassung mehrerer Felder erfordert gesonderte Maßnahmen zur Anpassung der in den Benutzerrealms abgespeicherten Benutzerdaten (siehe „Benutzerdaten anpassen").
- einen Vektor zu einem Feld vom Typ CHAR zusammenfassen
  Die Gesamtlänge muss dabei erhalten bleiben. Die Zusammenfassung eines Vektors erfordert gesonderte Maßnahmen zur Anpassung der in den Benutzerrealms abgespeicherten Benutzerdaten (siehe „Benutzerdaten anpassen").
- ein Feld vom Typ CHAR in einen Vektor umwandeln
  Die Gesamtlänge muss dabei erhalten bleiben. Die Erzeugung eines Vektors erfordert gesonderte Maßnahmen zur Anpassung der in den Benutzerrealms abgespeicherten Benutzerdaten (siehe „Benutzerdaten anpassen").
- eine Wiederholungsgruppe zu einem oder mehreren Feldern vom Typ CHAR zusammenfassen
  Die Gesamtlänge muss dabei erhalten bleiben. Die Zusammenfassung einer Wiederholungsgruppe erfordert gesonderte Maßnahmen zur Anpassung der in den Benutzerrealms abgespeicherten Benutzerdaten (siehe „Benutzerdaten anpassen").
- ein Feld oder mehrere hintereinanderliegende Felder vom Typ CHAR in eine Wiederholungsgruppe umwandeln
  Die Gesamtlänge muss dabei erhalten bleiben. Die Erzeugung einer Wiederholungsgruppe erfordert gesonderte

Maßnahmen zur Anpassung der in den Benutzerrealms abgespeicherten Benutzerdaten (siehe „Benutzerdaten anpassen").

- evtl. vorhandene freie Bytes vor einem implizit ausgerichteten numerischen Feld (FIXED BINARY) auf Stufe 0 für neue Felder nutzen
  In Wiederholungsgruppen sind FIXED BINARY Felder in den abgespeicherten Sätzen nicht ausgerichtet. Deshalb gibt es vor diesen Feldern keine impliziten freien Bytes.
- mehrere vorhandene Feldern in mehrere neue Felder neustrukturieren
  Für die Neustrukturierung müssen zwei RENAME/BALTER-Zyklen durchlaufen werden. Zunächst werden die Ausgangsfelder zu einem CHAR-Feld zusammengefasst. In einem zweiten Schritt kann dieses CHAR-Feld in die neue Struktur aufgeteilt werden. Diese Vorgehensweise kann besonders auch für das Zusammenfassen oder Aufteilen von NCHAR-Feldern genutzt werden.
- Felder vom Typ DBKEY bzw. DBKEY-LONG umbenennen
  Felder vom Typ DBKEY bzw. DBKEY-LONG, die mit LOCATION MODE DIRECT verwendet werden, können in einem Umbenennungszyklus nicht mit anderen Feldern zusammengefasst werden.
- Satzarten umbenennen
  Die Umbenennung muss in DDL und SSL an allen Stellen erfolgen.
- Sets umbenennen
  Die Umbenennung muss in DDL und SSL an allen Stellen erfolgen.
- Realms umbenennen
  Die Umbenennung muss in DDL und SSL an allen Stellen erfolgen. Ein alter Realmname darf in einem Umbenennungsschritt nicht sofort als neuer Name für einen anderen Realm genutzt werden. Es darf keine Datei mit dem neuen Dateinamen des User Realms existieren.

7.2 Schema-SSL ändern

Alle Namensänderungen der Schema-DDL müssen in der Schema-SSL übernommen werden.

Auch wenn Sie Ihre bisherige SSL unverändert verwenden können, müssen Sie diese neu übersetzen, da BGSIA sonst die Standardwerte der Speicherstruktur einsetzt!

Folgende Änderungen der Speicherstruktur sind zugelassen:

- Feldnamen in Satzarten ändern
  Die Namen müssen an allen Stellen (z.B. auch bei Schlüsseldefinitionen) geändert werden.
- Felder vom Typ DBKEY bzw. DBKEY-LONG umbenennen
  Felder vom Typ DBKEY bzw. DBKEY-LONG, die mit LOCATION MODE DIRECT verwendet werden, können in einem Umbenennungszyklus nicht mit anderen Feldern zusammengefasst werden.
- Satzarten umbenennen
  Die Umbenennung muss in DDL und SSL an allen Stellen erfolgen.
- Sets umbenennen
  Die Umbenennung muss in DDL und SSL an allen Stellen erfolgen.
- Realms umbenennen
  Die Umbenennung muss in DDL und SSL an allen Stellen erfolgen. Ein alter Realmname darf in einem Umbenennungsschritt nicht sofort als neuer Name für einen anderen Realm genutzt werden. Es darf keine Datei mit dem neuen Dateinamen des User Realms existieren.

7.3 Sicherungsmaßnahmen und Verhalten im Fehlerfall

Eine Umbenennung verändert im Allgemeinen nur die Compilerdatenbank. Dafür kann das After-Image-Logging eingeschaltet bleiben.

Lediglich bei der Umbenennung von Benutzerrealms erfolgen im Umbenennungsprozess in den Benutzerrealms kleine Änderungen. Dafür muss das After-Image-Logging ausgeschaltet werden.

7.3.1 Datenbank sichern

Falls es bei der Umbenennung zu einem Fehler kommt, muss die Datenbank in den Zustand vor dem Beginn des Umbenennungsprozesses zurückgesetzt werden. Dazu gibt es folgende Möglichkeiten:

- Einspielen einer Sicherung der Datenbank und Nachfahren der After-Image-Logging-Dateien bis zum letzten Konsistenzpunkt vor der Umbenennung.
- Verwendung einer Sicherung, die direkt vor dem Umbenennungszyklus erstellt wurde. Darin müssen folgende Dateien gesichert sein:
  - *dbname*.COSSD
  - *dbname*.DBDIR
  - *dbname*.DBCOM
  - Benutzerrealms, die umbenannt werden sollen.

Weitere Informationen zur Sicherung einer Datenbank finden Sie im Handbuch „Datenbankbetrieb“, Datenbank sichern und wiederherstellen im Fehlerfall.

7.3.2 Datenbank rekonstruieren

Bricht während des Umbenennungsprozesses ein Programm die Verarbeitung mit „ABNORMAL END“ ab, so müssen Sie, je nach Schwere des aufgetretenen Fehlers und Stelle innerhalb des Umbenennnungszyklus eine der folgenden Maßnahmen durchführen:

- die Ausführung des abgebrochenen Programms wiederholen
- auf die angelegte Sicherung zurückgreifen und den Umbenennungsprozess wiederholen

Wann es erforderlich ist, auf eine Sicherung der Datenbank zurückzugreifen und den Umbenennungsprozess zu wiederholen und wann es genügt, das abgebrochene Programm zu wiederholen, ist bei den einzelnen Programmen beschrieben!

Folgende Tabelle zeigt, welche Programme im Laufe der Umbenennung Dateien oder Realms der Datenbank verändern:

	DBDIR	**DBCOM**	**DBCOM.O**	**COSSD**	**COSSD.O**	**Benutzerrealms, auf die zugegriffen werden muss**
BRENAME	LS	LS	S	L	S	-
DDL-Compiler	LS	LS	-	S	-	-
SSL-Compiler	LS	LS	-	S	-	-
BGSIA	LS	LS	-	-	-	-
BALTER (Umbenennungsphase)	LS	L	L	-	-	LS
DDL-Compiler (Subschemata)	LS	LS	-	S	L	-
BGSSIA	LS	L	-	-	-	-

Tabelle 47: Zugriff auf die Dateien und Realms der Datenbank während einer Umbenennung

L lesender Zugriff

S schreibender Zugriff

- kein Zugriff

Für die Rekonstruktion der Datenbank stehen Ihnen folgende Möglichkeiten zur Verfügung:

- Sie können die Schattendatenbank in eine Originaldatenbank umwandeln durch Umbenennen mit dem MODIFY-FILE-ATTRIBUTES-Kommando.
- Sie können die ARCHIVE-Sicherung einspielen und evtl. den Datenbanknamen ändern mit dem MODIFY-FILE-ATTRIBUTES-Kommando. Wenn die ARCHIVE-Sicherung online erstellt wurde, müssen Sie sie evtl. mit dem Dienstprogramm BMEND nachfahren (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BMEND).

Weitere Informationen zur Rekonstruktion einer Datenbank finden Sie im Handbuch „Datenbankbetrieb“, Datenbank sichern und wiederherstellen im Fehlerfall.

7.4 Umbenennung einleiten mit BRENAME

Die Aufgabe von BRENAME beim Umbenennen einer Datenbank ist vergleichbar mit der Aufgabe von BCREATE beim Einrichten einer Datenbank: BRENAME bereitet die Compilerdatenbank auf die Aufnahme des neuen Schemas vor. Im Einzelnen führt BRENAME folgende Vorarbeiten zum Umbenennen aus:

- Sicherstellen der alten SIA im DBDIR und das DBDIR auf die Aufnahme einer neuen SIA vorbereiten, sodass im DBDIR nach dem BGSIA-Lauf für das neue Schema eine neue und eine alte SIA gespeichert sind. BALTER benötigt beide SIAs beim Anpassen der Strukturdaten an das neue Schema, um die Abweichungen des neuen Schemas vom alten Schema erkennen zu können.
  Achten Sie daher darauf, dass vor dem BRENAME-Lauf genügend freie Seiten im DBDIR zur Verfügung stehen bzw. durch Sekundärzuweisung > 0 eine automatische Realmerweiterung möglich ist.
- Löschen aller Benutzer-SSIAs im DBDIR.
- Sicherstellen des alten DBCOM in der Datei *dbname*.DBCOM.O und Neuformatieren des DBCOM.

  BALTER benötigt die Schema-Informationen des alten und des neuen DBCOM zum Untersuchen der geplanten Umbenennungen.
- Sicherstellen des alten COSSD in der Datei *dbname*.COSSD.O.

  Der DDL-Compiler benötigt nach der Umbenennung den alten COSSD für das Übernehmen der kompatiblen Subschemata. Es wird daher empfohlen, die Datei *dbname*.COSSD.O solange zur Verfügung zu halten, bis nach der Umbenennung alle Subschemata übersetzt sind.

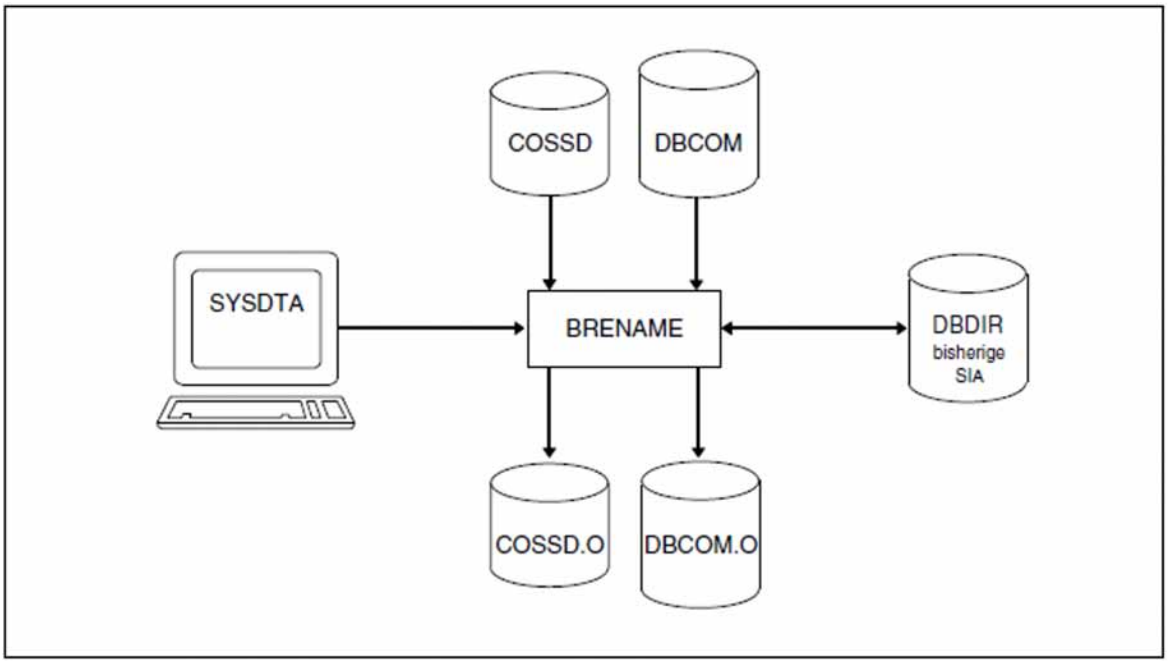

Bild 34: Systemumgebung beim Vorbereiten der Compilerdatenbank

BRENAME legt die Kopien des DBCOM und des COSSD automatisch auf gemeinschaftlicher Platte an. Ein CREATE-FILE-Kommando vor dem Starten von BRENAME zum Einrichten der beiden Dateien ist nur erforderlich, wenn die Kopien auf privater Platte angelegt werden sollen.

Je nach Größe der Dateien ist es allerdings ratsam, sie - auch wenn sie auf gemeinschaftlicher Platte liegen sollen - mit einem CREATE-FILE-Kommando mit SPACE-Angabe einzurichten (siehe auch „Maximalgröße für UDS/SQL-Dateien“, "Dateien und Realms einer UDS/SQL-Datenbank").

BRENAME erweitert bei Bedarf automatisch die Realms der bearbeiteten Datenbank. Näheres hierzu siehe Handbuch „Datenbankbetrieb", Automatische Realm-Erweiterung durch Dienstprogramme).

BRENAME berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

Kommandofolge zum Starten von BRENAME

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

Das Dienstprogramm BRENAME starten Sie in der Kennung, unter der die Datenbank katalogisiert ist, mit folgenden Kommandos:

```
01  [/CREATE-FILE FILE-NAME=dbname.DBCOM.0 ...]
02  [/CREATE-FILE FILE-NAME=dbname.COSSD.0 ...]
03  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
04  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
05  /START-UDS-BRENAME
06  END
```

01,02 Siehe Abschnitt „Compilerdatenbank einrichten".

04 Die angegebene Version von BRENAME wird ausgewählt.

Die Angabe der Version wird generell empfohlen, da mehrere UDS/SQL-Versionen parallel installiert sein können.

05 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen BRENAME gestartet werden.

06 BRENAME wird beendet.

Die END-Anweisung ist die einzige BRENAME-Anweisung.

7.5 Schema-DDL übersetzen

Wenn Sie mit dem Dienstprogramm BRENAME die Compilerdatenbank für die Aufnahme eines neuen Schemas vorbereitet haben, so müssen Sie als Nächstes mit dem DDL-Compiler Ihre Schema-DDL mit den neuen Namen übersetzen.

Der Übersetzungsvorgang entspricht dem beim Einrichten der Datenbank.

Nach dem Übersetzen der Schema-DDL existieren:

- ein alter und ein neuer DBCOM
- eine alte SIA im DBDIR
- ein alter und ein neuer COSSD

Kommandofolge zum Übersetzen der aktuellen Schema-DDL

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

Die aufgeführten Kommandos sind im Abschnitt „Schema-DDL übersetzen" ausführlich beschrieben!

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-DDL
04  SOURCE IS 'schemadatei'
05  END
06  /ASSIGN-SYSDTA TO=*SYSCMD
```

> Achten Sie darauf, dass der DDL-Compiler die Übersetzung mit „NORMAL END“ abschließt.
> Wenn er sie mit „ABNORMAL END“ beendet, so müssen Sie die Übersetzung mit korrigierten DDL-Klauseln wiederholen.

7.6 SSL übersetzen

Wahlweise können Sie, nachdem Sie Ihre Schema-DDL übersetzt haben, mit dem SSL-Compiler eine neue SSL übersetzen.

Führen Sie keine SSL-Übersetzung durch, so werden die Standardwerte für die Speicherstruktur eingesetzt. Wollen Sie Ihre bisher definierte Speicherstruktur beibehalten, so müssen Sie Ihre ursprünglichen SSL-Klauseln mit den der DDL-Source entsprechenden Umbenennungen neu übersetzen!

Der Übersetzungsvorgang entspricht dem beim Einrichten der Datenbank.

Kommandofolge zum Übersetzen der SSL

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme". Dort finden Sie auch die möglichen Aliasnamen für den Aufruf).

Die aufgeführten Kommandos sind im Abschnitt „SSL übersetzen" ausführlich beschrieben!

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-SSL
04  SOURCE IS 'ssl-datei'
05  END
06  /ASSIGN-SYSDTA TO=*SYSCMD
```

> Achten Sie darauf, dass der SSL-Compiler die Übersetzung mit „NORMAL END“ abschließt. Wenn er die Übersetzung mit „ABNORMAL END“ beendet, so müssen Sie folgende Maßnahmen durchführen:
>
> - bei Fehlern in den SSL-Klauseln:
>   die fehlerhaften SSL-Klauseln korrigieren und die SSL-Übersetzung wiederholen;
> - bei Fehlern in den DDL-Klauseln:
>   - die fehlerhaften DDL-Klauseln korrigieren
>   - das fehlerhafte Schema löschen in einem DDL-Lauf mit der Anweisung DELETE SCHEMA *schemaname*
>   - den Umbenennungsprozess ab „Übersetzen der Schema-DDL“ wiederholen.

7.7 Neue SIA erzeugen und in das DBDIR eintragen mit BGSIA

Nach dem erfolgreichen Übersetzen der Schema-DDL und (wahlweise) der SSL müssen Sie mit dem Dienstprogramm BGSIA die SIA des neuen Schemas erzeugen und in das DBDIR eintragen.

Die gesicherte SIA des alten Schemas bleibt im DBDIR stehen, sodass das DBDIR nach dem BGSIA-Lauf die SIA des alten und des neuen Schemas enthält. BALTER benötigt beide zur Prüfung und Durchführung der geplanten Umbenennungen.

Der BGSIA-Lauf entspricht dem beim Einrichten der Datenbank (siehe „Schema Information Area (SIA) erzeugen mit BGSIA"). Das Modul UDSHASH, das BGSIA erzeugt, müssen Sie nach dem BGSIA-Lauf in die HASHLIB eintragen.

Damit BGSIA im Umbenennnungszyklus gleichbleibende Realms, Satzarten und Sets trotz Namensänderung erkennt und die vorhandenen Referenzen unverändert lässt, müssen Namensänderungen für Realms, Satzarten und Sets vollständig in zusätzlichen Anweisungen übergeben werden. Davon betroffen sind die Namen in der DDL /SSL-Source. Die zusätzlichen Anweisungen des BGSIA verändern keine Namen in diesen Sourcen.

Dynamische Sets können genauso umbenannt werden. Die Namen, der für die IQS-Nutzung benötigten dynamischen Sets sind fest vorgegeben und dürfen nicht umbenannt werden. Dies wird jedoch nicht geprüft. Implizite Sets müssen nicht explizit angegeben werden. Sie werden automatisch mit der entsprechend umbenannten Satzart (SYS_recordname) umbenannt, sobald ein Search-Key definiert ist.

Wenn Sie mit eigenen Hashroutinen arbeiten, so müssen Sie diese spätestens vor dem Starten von BALTER zusätzlich mit den Attributen RMODE=ANY und AMODE=ANY in die HASHLIB eintragen.

Wird bei der Erzeugung der SIA erkannt, dass nicht nur reine Umbenennungen vorliegen, wird eine Meldung ausgegeben, dass der Umbenennungszyklus abgebrochen wird, da die Referenzen nicht zusammenpassen. Diese Meldung kann zur Sourcekorrektur genutzt werden. Eine genaue Analyse erfolgt erst mit BALTER. Änderungen die einer reinen Umbenennung widersprechen, sind z. B. nicht erlaubte Typänderungen oder die nicht erlaubte Aufteilung bzw. Zusammenfassung von Feldern.

SIA erzeugen und in das DBDIR eintragen

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /DELETE-SYSTEM-FILE FILE-NAME=*OMF
04  /START-UDS-BGSIA
05  GENERATE SCHEMA schemaname
06  [DISPLAY[ SCHEMA schemaname]]
07  END
```

Das Modul UDSHASH in die HASHLIB eintragen

```
01 /START-LMS
02 //OPEN-LIB LIB=dbname.HASHLIB,MODE=*UPDATE
03 //ADD-ELEMENT FROM-FILE=*OMF,TO-ELEMENT=*LIBRARY-ELEMENT(TYPE=R)
04 //END
```

7.8 Umbenennung prüfen und Strukturinformationen aktualisieren mit BALTER

Im Umbenennungszyklus prüft das Dienstsprogramm BALTER ausschließlich, ob die vorgesehenen Änderungen sich wirklich nur auf Umbenennungen beschränken.

Die einzig erlaubte BALTER-Anweisung (neben der END-Anweisung) ist im Umbenennungszyklus die REPORT-Anweisung. Standardmäßig ist REPORT IS YES eingestellt. Alle anderen BALTER-Anweisungen werden abgewiesen und führen zum Abbruch des BALTER-Laufes.

Ist REPORT IS YES eingestellt, startet BALTER nach der Analysephase die REPORT-Phase, in der er das Analyseprotokoll auf SYSLST ausgibt (siehe Abschnitt „Beschreibung des Analyseprotokolls (REPORT-Phase)"). Im Analyseprotokoll listet BALTER alle Änderungen auf, die während der Umbenennungsphase an Satzarten, Sets oder Schlüsseln vorgenommen werden. Ebenso enthält das Analyseprotokoll Fehlermeldungen und Warnungen über nicht zugelassene Änderungen (siehe Abschnitt „Beschreibung der BALTER-Meldungen").

REPORT IS NO sollten Sie nur nutzen, wenn sichergestellt ist, dass die Umbenennung erfolgreich durchgeführt werden kann. Ist dies nicht der Fall, wird die Diagnose von Fehlern im Schema erschwert.

BALTER können Sie nur dann erfolgreich ablaufen lassen, wenn Sie zuvor mit dem Dienstprogramm BGSIA die neue SIA erzeugt und ins DBDIR eingetragen haben (siehe Abschnitt „Neue SIA erzeugen und in das DBDIR eintragen mit BGSIA"). Andernfalls beendet sich der BALTER-Lauf mit der Meldung „BGSIA HAS NOT BEEN EXECUTED“.

Mit Hilfe des alten und des neuen DBCOM bzw. der alten und der neuen SIA stellt BALTER die Unterschiede zwischen der alten und der neuen Schema-Beschreibung fest und prüft, ob nur Umbenennungen vorgenommen wurden. Falls Umstrukturierungen in den Benutzerdaten notwendig wären, weil Datenbanksätze (z.B. Set Connection Data) und Tabellen (Adresslisten, Listen, SEARCH-Key-Tabellen usw.) geändert werden, wird der Umbenennungszyklus mit einer Fehlermeldung abgebrochen. Gegebenenfalls muss dann auf den Stand der Datenbank vor dem Umbenennungszyklus zurückgesetzt werden.

Änderungen, bei denen zwar keine Datenbanksätze oder Tabellen geändert werden, die vorhandenen Daten aber eventuell nicht ohne Weiteres weiter genutzt werden können, werden mit einer entsprechenden Warnung zugelassen. Solche Änderungen könnten Typänderungen oder das Zusammenfassen und Aufteilen von Feldern sein (siehe auch Abschnitt „Benutzerdaten anpassen").

7.8.1 Kommandofolge zum Starten von BALTER

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

Sie können BALTER unter der Kennung, in der die Datenbank katalogisiert ist, mit folgendem Kommando starten (Sie können das Programm auch mit dem Aliasnamen BALTER starten):

```
01  /START-UDS-BALTER
02  REPORT IS YES.
03  END.
```

7.8.2 Beschreibung der BALTER-Prüfung

BALTER prüft die Umbenennung in einer vorgegebenen Reihenfolge.

BALTER beginnt mit der Prüfung grundlegender Strukturen der Schemata ("SIA_CONTROL"). Hierzu gehören die Anzahl von Realms, Satzarten, Sets und Schlüsseln im alten und neuen Schema. Bezüglich dieser Anzahlen dürfen keine Änderungen vorliegen.

Anschließend werden die Area-Deklarationen geprüft. Hier sind nur Umbenennungen von Benutzerrealms zulässig.

Danach werden die Deklarationen der Satzarten geprüft. Hierbei werden die Feldumbenennungen, die Feldzusammenfassungen und die Feldaufteilungen geprüft. Ebenso wird geprüft, ob die Deklarationen von CALC-Key übereinstimmen.

Zuletzt werden die Set-Deklaration und damit auch die jeweils logisch zusammenhängenden Schlüssel-Deklarationen auf Übereinstimmungen überprüft. Hier sind nur Umbenennungen von Sets zulässig.

Zum Abschluss des BALTER-Laufs werden evtl. Umbenennungen von Realms durch Umkatalogisieren der entsprechenden Realms vollzogen. Dies ist der einzige Fall, bei dem im Rahmen eines Umbenennungszyklus Benutzerrealms geöffnet und bearbeitet werden müssen.

In der Listenausgabe werden alle Umbenennungen sowie Feldaufteilungen und Feldzusammensetzungen bei REPORT IS YES dokumentiert. Darüberhinaus wird ausgegeben, wieviele Realms, Satzarten (Records) und Sets unverändert sind.

Änderungen im Schema, die einer reinen Umbenennung widersprechen, führen zu Fehlermeldungen und zum Abbruch des Umbenennungszyklus. Dabei kann ein Fehler zu mehreren Fehlermeldungen führen, da sich an mehreren Stellen in der SIA relevante Änderungen ergeben können. Die Analyse wird nicht nach dem ersten erkannten Fehler beendet, sondern erst nach Auftreten mehrerer Fehler abgebrochen.

7.9 Unzulässige Schemaänderungen im Umbenennungszyklus

Um Verletzungen der Umbenennungsregeln schnell korrigieren zu können, sollte vor dem Umbenennungszyklus eine aktuelle Ausgabe des alten Schemas mit dem Dienstprogramm BPSIA erstellt werden. Ebenso sollten Sie im Umbenennungszyklus bei dem Dienstprogramm BGSIA die Anweisung DISPLAY nutzen. In einigen Fällen erkennt bereits BGSIA, dass bei der Umbenennung widersprechende Deklarationen im neuen Schema enthalten sind.

Im Umbenennungszyklus bezieht sich die Ausgabe von BALTER, über nicht erlaubte Unterschiede zwischen altem und neuem Schema, großteils auf die entsprechenden Informationen in der Ausgabe von BPSIA und BGSIA. Ebenfalls gibt BALTER die Unterschiede von einzelnen Feldern von Satzarten aus.

> ¡ Bei einer Korrektur sollten zunächst die Hinweise bei den Satzarten beachtet werden, da Meldungen zu unzulässigen Änderungen bei SIA_CONTROL, den Sets und Schlüsseln sehr häufig auf unzulässige Deklarationen bei Feldern der Satzarten zurückzuführen sind.

Auf unzulässige Schemaänderungen wird in fünf Phasen geprüft.

In Phase eins kontrolliert BALTER die SIA-CONTROL. Veränderungen in der SIA-CONTROL deuten auf grundsätzlich unzulässige Veränderungen hin. Folgende unzulässige Änderungen können auftreten (Meldungstext: DIFFERENCE IN sia-inhalt):

Meldung (sia-inhalt)	**Bedeutung**
NR_AREAS	Die höchste vergebene Realm-Nummer (Area Referenz) ist im alten und neuen Schema unterschiedlich.
NR_RECORDS	Die höchste vergebene Satzartnummer (REC-REF) ist im alten und neuen Schema unterschiedlich.
NR_SETS	Die höchste vergebene Setnummer (SET-REF) ist im alten und neuen Schema unterschiedlich.
NR_KEYS	Die höchste vergebene Schlüsselnummer (KEY-REF) ist im alten und neuen Schema unterschiedlich.
SCHEMA_NAME	Der alte und der neue Schemaname stimmen nicht überein.
`IMPL_RESULT_SET`	Es gibt Unterschiede beim IMPLICIT_RESULT_SET des Schemas.
`SINGULAR_SET`	Es gibt Unterschiede beim ersten SYSTEM-Set des Schemas.
`DYNAMIC_SET`	Es gibt Unterschiede beim ersten dynamischen Set des Schemas.
`MAX_REC_LENGTH`	Die Länge der größten Satzart ist im alten und neuen Schema unterschiedlich.
`MAX_ENTRY_LENGTH`	Die Länge der längsten Schlüssel ist im alten und neuen Schema unterschiedlich.
`MAX_MEMBERSHIPS`	Die größte Anzahl von Sets, in denen eine Satzart Member ist, ist im alten und neuen Schema unterschiedlich.
`MAX_SPLIT_PARAMETER`	Die Maximalzahl der in der SSL beim Parameter REORGANIZATION festgelegten Seiten ist unterschiedlich.
`LENGTH_KEY_BIT`	Die Kontrollinformation für die MODIFY-Information unterscheidet sich im alten und neuen Schema.

BLOCK_LENGTH	Die Länge der Datenbankseiten ist im alten und neuen Schema unterschiedlich.

In Phase zwei werden die einzelnen Areas auf unzulässige Änderungen geprüft. Folgende unzulässige Änderungen können auftreten (Meldungstext: DIFFERENCE IN area-inhalt):

Meldung (area-inhalt)	**Bedeutung**
AREA_REF	Die Referenz des Realms ist im alten und neuen Schema unterschiedlich.
AREA_PROPERTIES	Die zentralen Eigenschaften des Realms unterscheiden sich im alten und neuen Schema (TEMP, D/T).
NR_WITHIN_RECORDS	Die Anzahl der Satzarten, die im Realm gespeichert werden können, ist im alten und neuen Schema unterschiedlich.
RECORD_LIST	In den Referenzen der Satzarten, die im Realm gespeichert werden können, gibt es Unterschiede im alten und neuen Schema.

Die Meldung DIFFERENT USE OF AREA-REF zeigt an, dass die Belegung dieser Realm-Referenz im neuen und alten Schema so unterschiedlich ist, dass eine weitergehende differenzierte Analyse nicht sinnvoll ist.

Die Meldung AREA RENAMING WITH ALOG zeigt an, dass die Umbenennung von Areas nur erlaubt ist, wenn das After-Image-Logging ausgeschaltet ist. Falls die Datenbank mit After-Image-Logging betrieben wird, müssen Sie es vor dem Umbenennungszyklus ausschalten.

In Phase drei werden

- die einzelnen Satzarten auf unzulässige Änderungen geprüft.
- die Schlüssel auf Satzartebene auf Änderungen bezüglich der SIA-Daten geprüft.
- die Schlüsselinformationen geprüft, die erst später in die SSIA eingetragen werden.
- die Umbenennungen von Feldern geprüft.

Bei der Änderung der einzelnen Satzarten können folgende unzulässige Änderungen auftreten (Meldungstext: DIFFERENCE IN satz-inhalt):

Meldung (satz-inhalt)	**Bedeutung**
REC_REF	Die Referenz der Satzart ist im alten und neuen Schema unterschiedlich.
LOCATION_MODE	Es gibt Unterschiede bzgl. der Lage des DBKEY- bzw. DBKEY-LONG-Feldes, das für LOCATION MODE DIRECT genutzt wird.
REC_PROPERTIES	Die zentralen Eigenschaften der Satzart unterscheiden sich im alten und neuen Schema.
IMPLICIT_SET	Es gibt Unterschiede beim impliziten Set der Satzart.
OWNER_CHAIN	Es gibt Unterschiede beim ersten Set, in dem die Satzart Owner ist.
MEMBER_CHAIN	Es gibt Unterschiede beim ersten Set, in dem die Satzart Member ist.
RECORD_LENGTH	Die Länge des Satzes ist im alten und neuen Schema unterschiedlich.
SYSTEM_INFO	Die Länge der Set-Connection-Data ist im alten und neuen Schema unterschiedlich.

PHYSICAL_CALC_INFO	Die Lage der CALC-Buckets ist im alten und neuen Schema unterschiedlich.
DBTT_ENTRY_LENGTH	Die Länge der DBTT-Einträge ist im alten und neuen Schema unterschiedlich.
LOCATION_VIA	Die Art des Location Modus ist im alten und neuen Schema unterschiedlich.

Die Meldung DIFFERENT USE OF REC-REF zeigt an, dass die Belegung dieser Satzart-Referenz im neuen und alten Schema so unterschiedlich ist, dass eine weitergehende differenzierte Analyse nicht sinnvoll ist.

Bei der Änderung eines Schlüssels in einem Set können bezüglich der SIA-Daten folgende unzulässige Änderungen auftreten:

Meldung (satz-inhalt)	**Bedeutung**
KEY_LENGTH	Die Schlüssellänge ist im alten und neuen Schema unterschiedlich.
HASH	Die Referenz auf die Hash-Routine ist im alten und neuen Schema unterschiedlich.
DBTT_COLUMN	Der zum Schlüssel gehörende DBTT-Eintrag ist im alten und neuen Schema unterschiedlich.
KEY_PROPERTIES	Die Eigenschaften des Schlüssels sind im alten und neuen Schema unterschiedlich.

Bei den Schlüsselinformationen die erst später in die SSIA eingetragen werden, können folgende unzulässige Änderungen auftreten (Meldungstext: DIFFERENCE IN schluesselinhalt):

Meldung (schluessel-inhalt)	**Bedeutung**
KEY_LTH	Die Schlüssellänge ist im alten und neuen Schema unterschiedlich.
DBTT_COL_NR	Der zum Schlüssel gehörende DBTT-Eintrag ist im alten und neuen Schema unterschiedlich.
INDEX_AREA_REF	Die Referenz des Realms ist im alten und neuen Schema unterschiedlich.
CALC_PROCEDURE_NAME	Der Name der CALC-Routine ist im alten und neuen Schema unterschiedlich.
CALC_PROCEDURE_NR	Die Nummer der CALC-Routine ist im alten und neuen Schema unterschiedlich.
KEY_BITS	Die Eigenschaften des Schlüssels sind im alten und neuen Schema unterschiedlich.
KEY_REF_NR	Die (interne) Referenznummer ist im alten und neuen Schema unterschiedlich.
NR_BUCKETS	Die Größe des CALC-Bereiches ist im alten und neuen Schema unterschiedlich.
USER_KEY_TYPE	Der Feldtyp des Schlüssels ist im alten und neuen Schema unterschiedlich.
KEY_INDICATOR	Die Eigenschaften des Schlüssels sind im alten und neuen Schema unterschiedlich.
KEY_ITEM_DISPL	Ein Schlüsselfeld hat im alten und neuen Schema einen unterschiedlichen Beginn.
KEY_ITEM_CONCAT	Die Zusammenfassung von Schlüsselfeldern im neuen Schema passt nicht zu den Schlüsselfeldern im alten Schema.

KEY_ITEM_SPLIT	Das Aufteilen von Schlüsselfeldern im neuen Schema passt nicht zu den Schlüsselfeldern im alten Schema.

Die Meldung DIFFERENT USE OF KEY-REF zeigt an, dass die Belegung dieser Schlüssel-Referenz im neuen und alten Schema so unterschiedlich ist, dass eine weitergehende differenzierte Analyse nicht sinnvoll ist.

Bei der Umbenennung von Feldern können folgende unzulässige Änderungen auftreten :

Meldung	**Bedeutung**
ITEM BOUNDARIES UNSUITABLE	Die Zusammenfassung oder Aufteilung von Feldern führt zu Überlappung.
CONCATENATION TO NON CHAR TYPE	Felder wurden zusammengefasst, aber das zusammengefasste Feld im neuen Schema ist nicht vom Typ CHAR.
SPLIT OF NON CHAR TYPE	Ein Feld wird aufgeteilt, aber das Feld im alten Schema ist nicht vom Typ CHAR.
FORM GROUP FROM NON CHAR TYPE	Felder wurden zu einer Gruppe zusammengefasst, aber nicht alle Felder im neuen Schema sind vom Typ CHAR.
SPLIT GROUP TO NON CHAR TYPE	Eine Gruppe wird in ein oder mehrere Felder aufgeteilt, aber nicht alle Felder im neuen Schema sind vom Typ CHAR.
LENGTH DIFFERENCE OF TYPE VARIABLE	Die maximale Länge eines variablen Feldes am Ende der Satzart wurde unzulässig verändert.
LENGTH DIFFERENCE OF LAST OLD ITEM	Die Länge des letzten Feldes im alten Schema weist auf eine unzulässige Umbenennung hin.
LENGTH DIFFERENCE OF LAST NEW ITEM	Die Länge des letzten Feldes im neuen Schema weist auf eine unzulässige Umbenennung hin.
DIFFERENCE IN COMPRESSED RECORD	In einer komprimierten Satzart sind unzulässige Veränderungen vorgenommen worden.
DIFFERENCE IN ITEM TYPE	Hinweis auf eine unzulässige Typänderung in Feld
TYPE CHANGE OF NON CHAR TYPE	Eine unzulässige Typänderung wurde festgestellt.
VECTOR CHANGE OF NON CHAR TYPE	Eine unzulässige Typänderung wurde bei der Änderung eines Vektors festgestellt.

In Phase vier werden

- die einzelnen Sets auf unzulässige Änderungen geprüft.
- die Schlüssel auf Setebene auf Änderungen bezüglich der SIA-Daten geprüft. Es können die gleichen unzulässigen Änderungen wie in Phase drei (siehe Phase drei) auftreten.

> ¡ Da die Schlüssel keine expliziten Namen haben, wird bei Satzarten und Sets eine entsprechende interne Nummerierung für verschiedene Schlüssel oder ein eindeutiger Name für den Schlüsseltyp ausgegeben. Unterschiede bei den Schlüsseln für Satzarten oder Sets resultieren aus unzulässigen Änderungen bei den entsprechenden Datenfeldern, aus denen die Schlüssel gebildet werden. Um diese Unterschiede zu beheben, müssen die Umbenennungsfehler korrigiert werden.

- die Schlüsselinformationen geprüft, die erst später in die SSIA eingetragen werden. Es werden die Schlüssel mit der entsprechenden Nummer ausgegeben, die auch im BPSIA-Protokoll des Schemas ausgegeben werden. Es können die gleichen unzulässigen Änderungen wie in Phase drei (siehe Phase drei) auftreten.

Bei der Änderung der einzelnen Sets können folgende unzulässige Änderungen auftreten (Meldungstext: DIFFERENCE IN set-inhalt):

Meldung (set-inhalt)	**Bedeutung**
`SET_REF`	Die Referenz des Sets ist im alten und neuen Schema unterschiedlich.
`SET_PROPERTIES`	Die Eigenschaften des Sets sind im alten und neuen Schema unterschiedlich.
`SET_MODE`	Der Set-Modus ist im alten und neuen Schema unterschiedlich.
`SET_ORDER`	Die Sortierreihenfolge ist im alten und neuen Schema unterschiedlich.
`OWNER_OF_SET`	Die Referenz des Owners ist im alten und neuen Schema unterschiedlich.
`MEMBER_OF_SET`	Die Referenz des Members ist im alten und neuen Schema unterschiedlich.
`OWNER_CHAIN`	Die Ownerverkettung ist im alten und neuen Schema unterschiedlich.
`MEMBER_CHAIN`	Die Memberverkettung ist im alten und neuen Schema unterschiedlich.
`SINGULAR_SET_CHAIN`	Die Verkettung der SYSTEM-Sets ist im alten und neuen Schema unterschiedlich.
`DYNAMIC_SET_CHAIN`	Die Verkettung der dynamischen Sets ist im alten und neuen Schema unterschiedlich.
`KEY_CHAIN`	Die Verkettung des Sets ist im alten und neuen Schema unterschiedlich.
`SET_MEMBERSHIP`	POPULATION- bzw. INCREASE-Klausel sind im alten und neuen Schema unterschiedlich.
`SET_CONNECTION`	Es gibt Unterschiede in den Set Connection Data.
`ANCHOR_DBKEY`	Der Database Key des Ankers eines singulären Sets ist im alten und neuen Schema unterschiedlich.

Die Meldung DIFFERENT USE OF SET-REF zeigt an, dass die Belegung dieser Set-Referenz im neuen und alten Schema so unterschiedlich ist, dass eine weitergehende differenzierte Analyse nicht sinnvoll ist.

In der fünften Phase wird die physische Reihenfolge aller Schlüssel der SIA überprüft die an Satzarten oder Sets hängen. Die physische Reihenfolge der Schlüssel muss unverändert sein, da ihnen eine implizite Nummerierung zugeordnet ist, welche die Datenbank nutzt. Hier können nochmals unzulässige Änderungen (siehe Phase drei) auftreten.

7.10 Subschemata anpassen

Beim Vorbereiten der Compilerdatenbank auf die Umbenennung löscht BRENAME unter anderem alle SSIA’s im DBDIR und alle Subschema-Informationen im DBCOM; der DDL-Compiler bereitet anschließend beim Übersetzen der neuen Schema-DDL den COSSD für die Aufnahme neuer Subschema-Informationen vor. Nach der Umbenennungsphase sind also alle alten Subschema-Informationen gelöscht; im neuen COSSD sind noch keine Subschema-Informationen eingetragen.

Daher müssen Sie, nachdem BALTER Ihre Datenbank umbenannt hat, alle Subschemata neu übersetzen, jeweils eine neue SSIA erzeugen und diese im DBDIR eintragen.

7.10.1 Kompatible Subschemata übernehmen

Oft sind nicht alle Subschemata von den Änderungen des Schemas betroffen. Daher kopiert BRENAME zu Beginn der Umbenennung den COSSD in die Datei COSSD.O, sodass trotz Umbenennung noch alle alten Subschema-Informationen vorhanden sind. Wollen Sie diese alten Subschemata übernehmen, so müssen Sie, nachdem BALTER die Umbenennungsphase erfolgreich abgeschlossen hat, mit dem DDL-Compiler einen Lauf zum Übernehmen der alten Subschemata durchführen.

Bei dem Compilerlauf zum Übernehmen der Subschemata liest der DDL-Compiler aus der Datei COSSD.O alle alten Subschemata aus und prüft sie durch Neuübersetzen auf Verträglichkeit mit dem neuen Schema. Dabei unterscheidet er drei Fälle:

- die alte Subschema-Beschreibung ist mit dem neuen Schema unverträglich
- das alte Subschema ist auf Grund von logischen und/oder physischen Veränderungen im Schema mit dem neuen Schema unverträglich, d.h. die Ausführung von DML-Anweisungen ist betroffen
- das alte Subschema ist von Änderungen im neuen Schema nicht betroffen

In den ersten beiden Fällen legt der DDL-Compiler weder im DBCOM noch im COSSD Subschema-Information ab. Nur im letzten Fall, wenn ein Subschema nicht von den Schemaänderungen betroffen ist, übernimmt er es aus der Datei COSSD.O, compiliert es neu und trägt die Subschema-Information in den neuen DBCOM und in den neuen COSSD ein. Für jedes übernommene Subschema müssen Sie anschließend mit dem Dienstprogramm BGSSIA eine neue SSIA generieren und diese im DBDIR eintragen.

Beachten Sie bitte, dass "Verträglichkeit" nur bedeutet, dass die Sicht des alten Subschemas auf das neue Schema die gleiche geblieben ist wie auf das alte Schema. Es bedeutet nicht, dass etwa bei Verwenden der "COPY [ALL] RECORD[S]"-Klausel automatisch die Sicht auf die (aufwärtskompatiblen) Änderungen im neuen Schema gewährt wird; möchten Sie diese erlangen, müssen Sie das Subschema neu übersetzen.

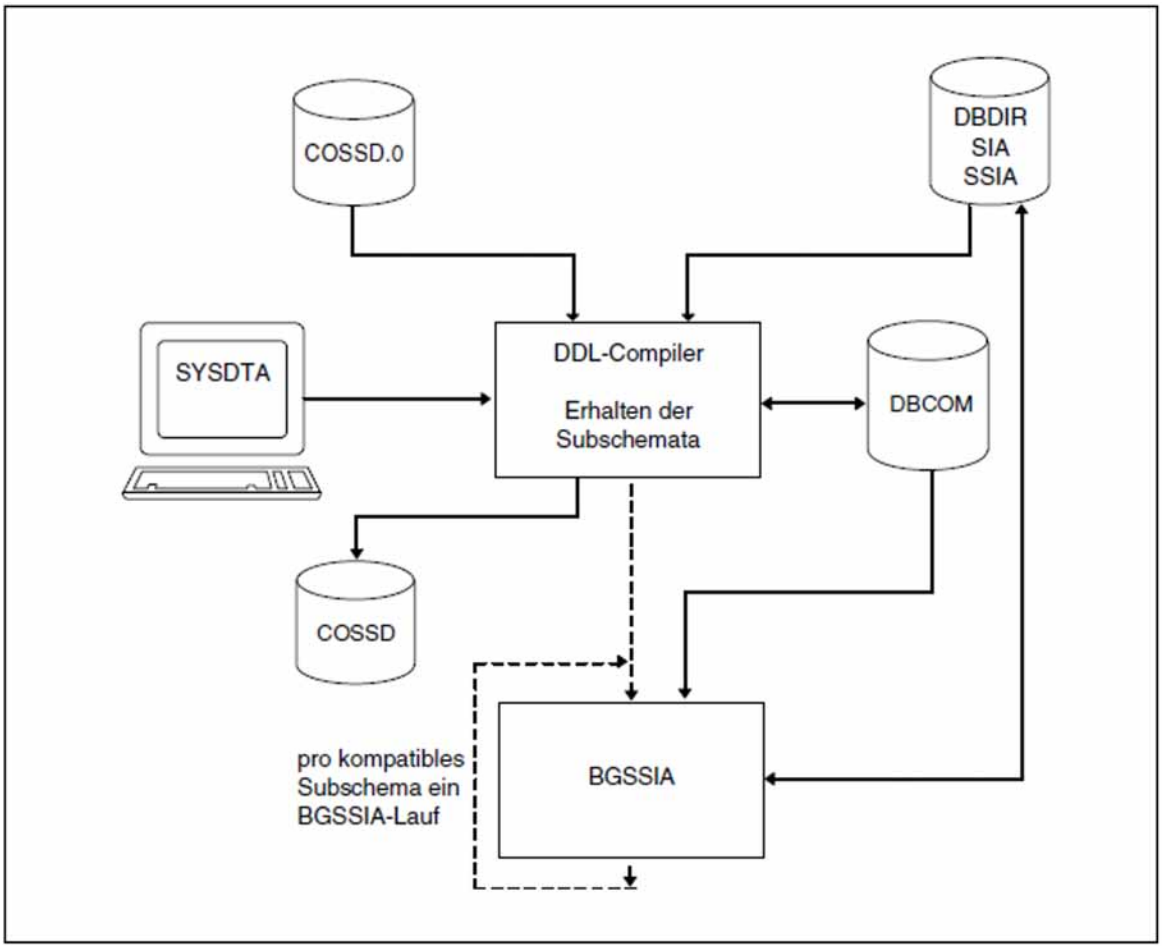

Bild 35: Systemumgebung beim Übernehmen der Subschemata

Den Compilerlauf zum Übernehmen der kompatiblen Subschemata können Sie wahlweise durchführen; lassen Sie ihn weg, so müssen Sie alle Subschemata einzeln neu übersetzen und die jeweilige SSIA neu generieren und in das DBDIR eintragen.

Verträglichkeiten/Unverträglichkeiten der Subschemata

schemaname und PRIVACY LOCK FOR COPY.....

> eine Änderung des Schemanamens und der PRIVACY LOCK-Angaben wirkt sich nicht auf das Übernehmen der Subschemata aus!
> Diese Änderungen müssen Sie erst bei nachfolgenden Subschema-Übersetzungen berücksichtigen.

PRIVACY LOCK FOR COMPILE

> bei dem Compilerlauf zum Übernehmen der Subschemata kopiert der DDL-Compiler diese PRIVACY-Angaben aus der **alten** Subschema-Beschreibung, sodass Zugriffssperren für das Übersetzen von Anwenderprogrammen erhalten bleiben.

bezeichner

> nicht verträglich ist ein altes Subschema mit dem neuen Schema, wenn Sie in der LOCATION MODE-Klausel, der WITHIN-Klausel (Satzartebene) oder der SET OCCURRENCE SELECTION-Klausel einen *bezeichner* hinzugefügt oder gelöscht oder den bisherigen *bezeichner* umbenannt haben, sofern die entsprechende Satzart bzw. der Set in dem Subschema liegt.

Anweisungen zum Übernehmen der Subschemata

Der DDL-Compiler benötigt zum Übernehmen der Subschemata folgende Anweisungen:

Anweisung	Standardwert	Bedeutung
COMPARE SUBSCHEMAS	-	Übernehmen der Subschemata veranlassen
[SORCLIST IS { YES | NO }]	YES	Listing der Subschemata ausdrucken
[DIAGNOSTIC { YES | NO }]	NO	Unverträglichkeiten der zum neuen Schema inkompatiblen Subschemata diagnostizieren und in Form von Fehlermeldungen auflisten
END	-	Eingabe der Anweisungen beenden

Tabelle 48: Anweisungen zum Übernehmen der Subschemata

Kommandofolge für das Übernehmen der Subschemata

Mit folgenden Kommandos können Sie einen DDL-Compilerlauf zum Übernehmen der Subschemata starten (siehe Abschnitt „Schema-DDL übersetzen"):

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-DDL
04  COMPARE SUBSCHEMAS
05  [DIAGNOSTIC {YES | NO}]
06  [SORCLIST IS {YES | NO}]
07  END
```

Für jedes übernommene Subschema müssen Sie mit folgenden Kommandos anschließend die SSIA erzeugen und im DBDIR eintragen (siehe Abschnitt „Subschema Information Area (SSIA) erzeugen mit BGSSIA"):

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-BGSSIA
04  GENERATE SUBSCHEMA subschemaname OF SCHEMA schemaname
05  [DISPLAY[ SUBSCHEMA subschemaname OF SCHEMA schemaname]]
06  END
```

7.10.2 Inkompatible Subschemata anpassen

Bei allen Subschemata, die in ihrer ursprünglichen Form mit dem neuen Schema nicht verträglich sind, müssen Sie folgende Maßnahmen durchführen:

- die Source bzgl. der Umbenennungen anpassen
- eventuell die Subschema-Beschreibung korrigieren
- mit dem DDL-Compiler das korrigierte Subschema neu übersetzen
- mit BGSSIA eine neue SSIA generieren und im DBDIR eintragen
- eventuell Anwenderprogramme neu übersetzen und binden

Die hier beschriebenen Kommandofolgen geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme". Dort finden Sie auch die möglichen Aliasnamen für den Aufruf).

Kommandofolge zum Anpassen der Subschemata

Korrigiertes Subschema übersetzen: (siehe Abschnitt „Schema-DDL übersetzen")

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-DDL
04  [DELETE 'subschemaname':'neuer schemaname']
05  SOURCE IS 'subschemadatei'
06  SORCLIST IS YES
07  END
08  /ASSIGN-SYSDTA TO=*SYSCMD
```

02 Versionsabhängiges Modul des linked-in DBH der entsprechenden Version wird nachgeladen (siehe Handbuch „Anwendungen programmieren“, Abschnitt „UDS/SQL-TIAM-Anwenderprogramme binden, laden und starten“).

03 Das UDS/SQL-Dienstprogramm kann auch mit dem Aliasnamen DDL gestartet werden.

04 Die Anweisung DELETE dürfen Sie nur angeben, wenn Sie ein Subschema geändert haben und neu übersetzen wollen, das der DDL-Compiler beim Übernehmen der Subschemata als kompatibel erkannt und bereits übernommen hatte.

SSIA erzeugen und in das DBDIR eintragen

Siehe Abschnitt „Subschema Information Area (SSIA) erzeugen mit BGSSIA".

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-BGSSIA
04  GENERATE SUBSCHEMA subschemaname OF SCHEMA schemaname
05  [DISPLAY[ SUBSCHEMA subschemaname OF SCHEMA schemaname]]
06  END
```

7.11 DB-Anwendungen anpassen

Eine sofortige Anpassung der DB-Anwendungen bzgl. neuer Namen ist nicht notwendig, wenn reine Umbenennungen ohne Typänderungen oder Aufteilung bzw. Zusammenfassung von Feldern vorgenommen wurde. In diesen Fällen sind Anpassungen erst bei der nächsten Übersetzung des Anwenderprogramms notwendig.

Anwendungen, die von den Umbenennungen nicht betroffen sind, müssen nicht geändert werden, z.B. wenn das Subschema die umbenannten Teile nicht enthält.

SSITAB-Module, die von den Umbenennungen nicht betroffen sind, müssen nicht mit BCALLSI neu erstellt werden.

Bei Änderung von Feldtypen bzw. beim Aufteilen oder Zusammenfassen von Feldern müssen alle betroffenen Anwendungen auf der Basis der neuen Subschemata neu übersetzen werden.

7.12 Zugriffsberechtigungen aktualisieren

Der Umbenennungsprozess ändert nichts an den Zugriffsberechtigungen, die Sie mit dem Dienstprogramm BPRIVACY eingetragen haben. Aufgrund der Umbenennungen von Satzarten, Sets und Benutzerrealms können allerdings Eingabedateien für die Vergabe von Zugriffsrechten nicht mehr aktuell sein. Sie sollten deshalb, nach der Umbenennung mit BPRIVACY bzw. ONLINE-PRIVACY und der Anweisung SHOW-USER-GROUP, eine Ausgabe der aktuellen Rechte erzeugen und die Eingabedateien entsprechend anpassen.

7.13 Benutzerdaten anpassen

Im Umbenennungszyklus werden bei Typänderungen oder der Aufteilung bzw. Zusammenfassung von Feldern nur die Schema- und Subschemadaten geändert. Die eigentlichen Benutzerdaten bleiben unverändert. Eine eventuell notwendige Initialisierung oder anderweitige Versorgung der entsprechend veränderten Datenfelder ist unabhängig vom Umbenennungszyklus durchzuführen. Dies müssen Sie vor oder nach dem Umbenennungszyklus parallel zum normalen Betrieb durchführen, z.B. mit geeigneten Anwendungsprogrammen.

7.14 Maßnahmen vor Wiederaufnahme des DB-Betriebs

Falls vor dem Umbenennungszyklus das After-Image-Logging ausgeschaltet wurde, ist dadurch eine Logginglücke entstanden. Nach dem Umbenennungszyklus kann mit dem Dienstprogramm BMEND das After-Image-Logging wieder eingeschaltet werden (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BMEND). Danach muss erneut eine Sicherung der Datenbank erzeugt werden (siehe Handbuch „Datenbankbetrieb“, Datenbank sichern und wiederherstellen im Fehlerfall).

Die Dateien DBCOM.O und COSSD.O können Sie nun wieder löschen.

7.15 Beispiel

BRENAME-Lauf und neue Schema-DDL und SSL übersetzen.

Ein Reservefeld in der Satzart Kunde wird aufgeteilt in ein neues Unicode-Feld für die Kopie der Adresse und das verbleibende Reservefeld.

/START-UDS-BRENAME

```
***** START       BRENAME      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:10

***** THE FILE: :SQL2:$XXXXXXXX.VERS.DBCOM IS COPIED TO:
      :SQL2:$XXXXXXXX.VERS.DBCOM.O

***** THE FILE: :SQL2:$XXXXXXXX.VERS.COSSD IS COPIED TO:
      :SQL2:$XXXXXXXX.VERS.COSSD.O
1005 RENAMING SUCCESSFULLY INITIATED

***** DIAGNOSTIC SUMMARY OF BRENAME

               NO WARNINGS
               NO ERRORS
               NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :          90
***** NORMAL END   BRENAME      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:11
```

/START-UDS-DDL

```
***** START         DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:11
*   DDLCOMP: INPUT SYSTEMPARAMETERS
```

SOURCE IS 'S.VERS.DDL.RENAME'
END

```
*  DDLCOMP: READ SCHEMA/SUBSCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:11/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:11/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  DDLCOMP: START SCHEMA-PHASE
*  DDLCOMP: CHECK SCHEMA RULES
*  DDLCOMP: CHECK DATA ALLOCATION
*  DDLCOMP: SEMANTIC TEST
*  DDLCOMP: CYCLUS TESTS
*  DDLCOMP: ERROR DIAGNOSTIC
*  DDLCOMP: NO ERRORS IN SCHEMA-PHASE
*  DDLCOMP: CREATE FILE COSSD
*  DDLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:11/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                    658       2026         67        926         39
%  UDS0213 UDS NORMAL BEENDET MIT *************658 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:11/0YA2)
```

```
***** DIAGNOSTIC SUMMARY FOR DDL-SCHEMA KUNDENDATEI

              NO ERRORS
+++++          9 WARNINGS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   DDLCOMP      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:11
```

/START-UDS-SSL

```
***** START       SSLCOMP       (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:11
*  SSLCOMP: INPUT SYSTEMPARAMETERS
```

SOURCE IS 'S.VERS.SSL.NEU'
END

```
*  SSLCOMP: READ SSL-SCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:11/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:11/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  SSLCOMP: START SSL-PHASE
*  SSLCOMP: CHECK SSL RULES
*  SSLCOMP: SEMANTIC TEST
*  SSLCOMP: ERROR DIAGNOSTIC
*  SSLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:11/0YA2)
 0YA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                     127        251         60         34         23
%  UDS0213 UDS NORMAL BEENDET MIT *************127 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:11/0YA2)

***** DIAGNOSTIC SUMMARY FOR SSL - SCHEMA

             NO ERRORS
             NO WARNINGS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   SSLCOMP      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:11
```

/START-UDS-BGSIA

```
***** START       BGSIA         (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:11
```

GENERATE SCHEMA KUNDENDATEI
DISPLAY
END

```
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:11/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:11/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
```

```
ESTIMATE-REPORT

********** FOR USER-REALM     3 NAME IS : SACHRLM
     A SIZE OF          24 BLOCKS WAS ESTIMATED

********** FOR USER-REALM     4 NAME IS : VERSICHERTENRLM
     A SIZE OF         239 BLOCKS WAS ESTIMATED

********** FOR USER-REALM     6 NAME IS : TRANSPORTRLM
     A SIZE OF          24 BLOCKS WAS ESTIMATED
END OF ESTIMATE-REPORT
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:12/0BCV)
 0YA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                     569        778         60        183         29
%  UDS0213 UDS NORMAL BEENDET MIT *************569 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:12/0YA2)

***** DIAGNOSTIC SUMMARY OF BGSIA

               NO WARNINGS
               NO ERRORS
               NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   BGSIA        (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:12
```

/MODIFY-JOB-SWITCHES ON=(1,4)
/START-LMS
//MODIFY-LOGGING-PARAMETERS LOG=*MAX
//OPEN-LIBRARY LIB=VERS.HASHLIB,MODE=*UPDATE
//ADD-ELEMENT FROM-FILE=*OMF,TO-ELEM=*LIB-ELEM(TYPE=R),WRITE-MODE=*ANY

```
INPUT  OMF
OUTPUT LIBRARY= :SQL2:$XXXXXXXX.VERS.HASHLIB
        ADD UDSHASH AS (R)UDSHASH/@(0003)/2017-06-28 , OUTPUT REPLACED
```

//SHOW-ELEM-ATTR

```
INPUT  LIBRARY= :SQL2:$XXXXXXXX.VERS.HASHLIB
TYP NAME    VER (VAR#) DATE        NAME    VER (VAR#) DATE
(R) ADMIN## @   (0001) 2017-06-28   UDSHASH @   (0003) 2017-06-28
      2 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
```

//END
/MODIFY-JOB-SWITCHES OFF=(1)

Prüfung der Umbenennung mit BALTER

/START-UDS-BALTER

```
***** START       BALTER        (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:12
```

REPORT   IS YES .
END.

```
+++++ WARNING: 1091     1 CHANGE(S) OF ITEM TYPE
+++++ WARNING: 1096     1 CHANGE(S) WITH SPLIT OF ITEMS

NUMBER OF FILE ACCESSES:           0

***** DIAGNOSTIC SUMMARY OF BALTER

+++++          2 WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :         119
***** NORMAL END   BALTER      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:12
```

Listausgabe von BALTER

```
 RENAME CHECK    FOR SCHEMA                    KUNDENDATEI
 RENAME CHECK    SIA_CONTROL
 RENAME CHECK    AREA
 NO CHANGES IN   ALL                          4 AREA(S)
 RENAME CHECK    RECORD

|-------------------------------------------------------------------------------------------
|
 | DIFFERENCE IN         RECORD
KUNDE                                        |

|-------------------------------------------------------------------------------------------
|

|
|
 |   +++ WARNING +++     DIFFERENCE IN       ITEM_TYPE           NOT-
USED                     |

|
|
 |   ITEM
SPLIT                                                                                  |
 |       ITEM-NAME (OLD SIA)           LENGTH TYPE    |   ITEM-NAME (NEW SIA)   LENGTH
TYPE  |
 |       NOT-USED                         255 CHAR    |   ADRESSE-U               240
NCHAR  |
 |                                                    |   NOT-USED                 15
CHAR   |

|-------------------------------------------------------------------------------------------
|
 NO CHANGES IN   REMAINING                    3 RECORD(S)
 RENAME CHECK    SET
 NO CHANGES IN   ALL                         13 SET(S)
 RENAME CHECK    KEYS IN SIA
 NO CHANGES IN   ALL                          5 KEY(S)

+++++ WARNING: 1091     1 CHANGE(S) OF ITEM TYPE

+++++ WARNING: 1096     1 CHANGE(S) WITH SPLIT OF ITEMS

NUMBER OF FILE ACCESSES:            0


***** DIAGNOSTIC SUMMARY OF BALTER

+++++          2 WARNINGS
              NO ERRORS
          NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY

***** NR OF DATABASE ACCESSES  :          119
```

Übersetzung des Subschemas

/START-UDS-DDL

```
***** START       DDLCOMP     (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:12
*  DDLCOMP: INPUT SYSTEMPARAMETERS
```

SOURCE IS 'S.VERS.SUBDDL.NEU'
END

```
*  DDLCOMP: READ SCHEMA/SUBSCHEMA
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:12/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:12/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*  DDLCOMP: START SUBSCHEMA-PHASE
*  DDLCOMP: CHECK SUBSCHEMA RULES
*  DDLCOMP: CHECK DATA ALLOCATION
*  DDLCOMP: SUBCOPY
*  DDLCOMP: ERROR DIAGNOSTIC
*  DDLCOMP: NO ERRORS IN SUBSCHEMA-PHASE
*  DDLCOMP: WRITE SUBSCHEMA ON COSSD
*  DDLCOMP: NO ERRORS DETECTED
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:12/0YA2)
 0YA2: DATABASE NAME          DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                  1379      2613        75       639        47
%  UDS0213 UDS NORMAL BEENDET MIT ************1379 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:12/0YA2)

***** DIAGNOSTIC SUMMARY FOR DDL-SUBSCHEMA

               NO ERRORS
               NO WARNINGS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   DDLCOMP      (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:12
```

/START-UDS-BGSSIA

```
***** START       BGSSIA        (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:12
```

GENERATE SUBSCHEMA VERWALTUNG OF SCHEMA KUNDENDATEI
DISPLAY
END

```
%  UDS0215 UDS STARTET UDS/SQL V2.9 (LINKED-IN), DATE=2017-06-28 (ILL2038,11:33:12/0YA2)
%  UDS0746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT (ILL1746,11:33:12/0YA2)
0YA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.DEFAULT
0YA2: DEFAULT PUBSET: SQL2
0YA2: ----------------------------------------
*** SSIA GENERATION NORMALLY ENDED.
*GENERATION OF ITEM-TABLE AND NAME-TABLE STARTED.
*GENERATION OF ITEM-TABLE AND NAME-TABLE FINISHED.
%  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK  (ILL1758,11:33:12/0YA2)
```

```
 0YA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE PHYS WRITE
 0YA2: ------------------------------------------------------------------------
 0YA2: VERS                     802       1392         76        297         29
%  UDS0213 UDS NORMAL BEENDET MIT *************802 DML-STATEMENTS 2017-06-28
(ILLY033,11:33:12/0YA2)

***** DIAGNOSTIC SUMMARY OF BGSSIA

               NO WARNINGS
               NO ERRORS
               NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NORMAL END   BGSSIA       (UDS/SQL  V2.9  0000 )     2017-06-28   11:33:12
```

Die Umbenennung der Datenbankobjekte ist hiermit fertig.

8 Datenbank auf größeres Seitenformat umstellen (BPGSIZE)

Eine UDS/SQL-Datenbank kann mit folgenden Seitenformaten aufgebaut sein (siehe Abschnitt „Compilerdatenbank formatieren mit BCREATE"):

- 2-Kbyte-Format mit 2048 Byte Seitenlänge
- 4-Kbyte-Format mit 4000 Byte Seitenlänge
- 8-Kbyte-Format mit 8096 Byte Seitenlänge

Das Dienstprogramm BPGSIZE können Sie dazu verwenden, um

- eine Datenbank auf ein Format mit größerer Seitenlänge umzustellen (siehe Abschnitt „Datenbank mit BPGSIZE umstellen").
- bei unveränderter Seitenlänge den Speicherplatzbedarf für die Realms der Datenbank zu verringern
- für jeden Realm möglicherweise vorhandene FPA-Extents zu entfernen. BPGSIZE fasst die gesamte FPA zu einer nur aus der FPA-Basis bestehenden FPA zusammen.
- DBTTs zusammenzufassen. BPGSIZE fasst DBTT-Basis und DBTT-Extents zu einer neuen DBTT-Basis zusammen.

Im Einzelnen umfasst die Umstellung einer Datenbank und der zugehörigen Anwendungen folgende Schritte:

- Kriterien prüfen, die eine Umstellung der Datenbank auf ein größeres Format erfordern
- Datenbank mit dem Dienstprogramm BPGSIZE auf das größere Format umstellen
- Datenbankschema und Subschemata anpassen (Datenbank umstrukturieren)
- ggf. von der Datenbankumstellung betroffene Anwendungen ermitteln und anpassen; die Umstellung der Datenbank auf ein größeres Seitenformat erfordert jedoch nicht in jedem Fall die Anpassung von Anwenderprogrammen, die mit der Datenbank arbeiten.

Die Umstellung der Datenbank mit BPGSIZE, die Umstrukturierung der Datenbank und die Anpassung der Anwenderprogramme können Sie zeitlich entkoppelt durchführen. Zwischen den einzelnen Schritten der Umstellung ist normaler Datenbankbetrieb möglich.

Am Ende des Kapitels finden Sie zwei Beispiele, die den Umstellvorgang in unterschiedlichen Anwenderszenarien skizzieren; u.a. wird auch der in der Praxis am häufigsten auftretende Fall beschrieben.

8.1 Kriterien für die Umstellung

Für die Umstellung einer Datenbank auf ein größeres Seitenformat kann es folgende Gründe geben:

- Kapazitätsengpässe
- Verwendung von Database-Key-Werten des erweiterten Wertebereichs
- Nutzung der vollen UDS/SQL-Funktionalität

Datenbank aus Kapazitätsgründen umstellen

Die Umstellung einer 2-Kbyte-Datenbank auf das 4-Kbyte- oder 8-Kbyte-Format kann aus einem der folgenden Kapazitätsgründe notwendig werden (siehe Handbuch „Entwerfen und Definieren"):

- Zu mindestens einer Satzart der Datenbank sollen künftig mehr als 16 777 215 (=2^{24}-1) Sätze abgespeichert werden.
- Bei mindestens einer Satzart beträgt die Satzlänge mehr als 1700 Byte (Richtwert; die maximale Satzlänge in einer 2-Kbyte-Datenbank beträgt 2020 Byte).
- Die Datenbank verwendet mehr als 200 Satzarten (Richtwert; in einer 2-Kbyte-Datenbank können Sie maximal 253 Satzarten definieren).
- Die Datenbank soll künftig mehr als 123 Realms enthalten.
- Eine Multi-DB-Konfiguration wurde nur erstellt, um die Sätze einer Satzart auf mehr als eine Datenbank zu verteilen (auf Grund der Kapazitätsgrenzen).

Die Umstellung einer 4-Kbyte-Datenbank auf das 8-Kbyte-Format ist erforderlich, wenn für eine Satzart der Datenbank die in 4-Kbyte-Datenbanken zulässige Satzlänge von maximal 3968 Byte nicht mehr ausreicht.

Ob und inwieweit die genannten Kriterien für eine Datenbank erfüllt sind, entnehmen Sie dem SIA-Protokoll des Dienstprogramms BPSIA und der Satzart-Statistik des Dienstprogramms BSTATUS (siehe Handbuch „Sichern, Informieren und Reorganisieren").

2-Kbyte-Datenbank wegen Verwendung erweiterter Database-Key-Werte umstellen

Auch eine 2-Kbyte-Datenbank, bei der zunächst keine Kapazitätsengpässe auftreten, müssen Sie auf das 4-Kbyte- oder 8-Kbyte-Format umstellen, wenn ein Anwenderprogramm einen Database Key der Datenbank mit Database-Key-Werten des erweiterten Wertebereichs versorgt (REC-REF > 254 und/oder RSQ > 2^{24}-1, siehe Handbuch „Entwerfen und Definieren").

Bei Multi-DB-Programmen, d.h. bei Anwenderprogrammen, die auf mehrere Datenbanken zugreifen, ist daher zu prüfen, ob diese Anwenderprogramme Database-Key-Werte datenbankübergreifend verwenden. Wenn eine solche Anwendung Database-Key-Werte des erweiterten Wertebereichs aus einer 4-Kbyte- oder 8-Kbyte-Datenbank in einen Database Key einer 2-Kbyte-Datenbank überträgt, müssen Sie diese 2-Kbyte-Datenbank ebenfalls auf das 4-Kbyte- oder 8-Kbyte-Format umstellen und ggf. entsprechend umstrukturieren (siehe Abschnitt „Umgestellte Datenbank umstrukturieren").

Grundsätzlich können Sie die Datenbanken, die von der Nutzung erweiterter Database-Key-Werte betroffen sind, in beliebiger Reihenfolge umstellen. Es empfiehlt sich jedoch folgendes Vorgehen:

1. Datenbanken umstellen und ggf. umstrukturieren, die von Anwenderprogrammen künftig mit erweiterten Database-Key-Werten versorgt werden.
2. Datenbanken umstellen und ggf. umstrukturieren, aus denen Anwenderprogramme künftig erweiterte Database-Key-Werte in andere Datenbanken übertragen.

Auf diese Weise schließen Sie Fehler aus, die beim Übertragen erweiterter Database-Key-Werte in noch nicht umgestellte Datenbanken entstehen können.

2-Kbyte-Datenbank umstellen, um volle UDS/SQL-Funktionalität zu nutzen

Datenbanken im 4-Kbyte- oder 8-Kbyte-Format benötigen Sie außerdem, wenn Sie folgende Funktionen nutzen wollen:

- Datenbanken auf NK4-Pubsets
- BS2000-Zugriffsmethode FASTPAM für Dateien und Realms der Datenbank

8.2 Datenbank mit BPGSIZE umstellen

Mit dem Dienstprogramm BPGSIZE stellen Sie eine Datenbank auf ein Format mit gleicher oder größerer Seitenlänge um. Die umzustellende Datenbank kann die Originaldatenbank sein oder eine Schattendatenbank mit einem von NEW verschiedenen Kopienamen (Suffix). Die umzustellende Datenbank muss konsistent sein und wird von BPGSIZE nicht verändert. Die Konsistenz der umzustellenden Datenbank überprüfen Sie mit dem Dienstprogramm BCHECK (siehe Handbuch „Sichern, Informieren und Reorganisieren").

BPGSIZE erzeugt die umgestellte Datenbank als Schattendatenbank mit Kopienamen (Suffix) NEW. Während der Umstellung mit BPGSIZE können andere UDS/SQL-Dienstprogramme sowie UDS/SQL-Sessions nur lesend auf die Datenbank zugreifen.

Das Dienstprogramm BPGSIZE arbeitet ohne AFIM-Logging. Falls das AFIM-Logging für die umzustellende Datenbank eingeschaltet ist, meldet BPGSIZE dies während der Umstellung und gibt am Ende der Bearbeitung zusätzlich eine Warnung aus. Bei den umgestellten Datenbanken ist sowohl das AFIM-Logging als auch die Online-Sicherungsfähigkeit stets ausgeschaltet.

Damit Sie die umgestellte Datenbank im laufenden UDS/SQL-Betrieb einsetzen können, sind nach dem BPGSIZE-Lauf noch eine Reihe zusätzlicher Maßnahmen erforderlich (siehe Abschnitt „Umgestellte Datenbank für den DB-Betrieb vorbereiten").

BPGSIZE berücksichtigt beim Start ggf. eine zugewiesene UDS/SQL-Pubset-Deklaration (siehe Handbuch „Datenbankbetrieb", Pubset-Deklarations-Jobvariable). Eine fehlerhafte Zuweisung führt zum Programmabbruch.

8.2.1 Funktionen von BPGSIZE

Das Dienstprogramm BPGSIZE bietet folgende Funktionen:

- Umstellen einer 2-Kbyte-Datenbank in eine Datenbank (Kopiename NEW) im 4-Kbyte- oder 8-Kbyte-Format. Beim Umstellen auf die größeren Seiten erweitert BPGSIZE u.a. Systemdatenstrukturen wie z.B. SCD und Seitenindex und richtet so die Datenbank für die Aufnahme und Verwaltung von Database-Key-Werten mit einer REC-REF > 254 und/oder einer RSQ > 2^{24}-1 ein (siehe Handbuch „Entwerfen und Definieren").
- Umstellen einer 4-Kbyte-Datenbank in eine Datenbank (Kopiename NEW) im 8-Kbyte-Format.
- Umstellen in eine Datenbank (Kopiename NEW), die durch Komprimierung der Realms bei unveränderter Seitenlänge weniger Speicherplatz benötigt als die ursprüngliche Datenbank.
  Die Komprimierung ist möglich, da BPGSIZE beim Umstellen der Datenbank weder Lücken in den Seiten der ursprünglichen Datenbank noch leere Seiten in die umgestellte Datenbank übernimmt, eine vorgegebene Placement Optimization dabei allerdings berücksichtigt.
- Entfernen von FPA-Extents
  BPGSIZE bewirkt für jeden Realm, dass die Basis-Freiplatzverwaltungstabelle (FPA-Basis) und möglicherweise vorhandene FPA-Extents zu einer nur aus der FPA-Basis bestehenden FPA zusammengelegt werden. FPA-Extents können im Rahmen einer automatischen Realm-Erweiterung, einer Online-Realm-Erweiterung wie auch bei einer Realm-Erweiterung durch das Dienstprogramm BREORG entstehen (siehe Handbuch „Datenbankbetrieb", Online-Realm-Erweiterung).
- Entfernen von DBTT-Extents
  BPGSIZE fasst jede vorhandene DBTT zu einer nur aus der DBTT-Basis bestehenden DBTT zusammen. Eventuell existierende DBTT-Extents werden eliminiert.

Realms, für die mit dem DAL-Kommando ACT INCR die Online-Erweiterbarkeit aktiviert wurde, sind auch nach einem BPGSIZE-Lauf online erweiterbar.

8.2.2 Realms und Dateien

Ausgehend von den Realms der ursprünglichen Datenbank erzeugt BPGSIZE die Realms der umgestellten Datenbank. Hierzu benötigt BPGSIZE Arbeitsdateien.

8.2.2.1 Realms der umgestellten Datenbank

Im Standardfall legt BPGSIZE einen umgestellten Realm auf gemeinschaftlichem Datenträger in der Größe des ursprünglichen Realm an.
Mit dem BS2000-Kommando CREATE-FILE können Sie die Datei für einen umgestellten Realm auch gezielt unter einer bestimmten Katalogkennung oder auf privater Platte anlegen.

Dateiname des umgestellten Realm

Der Dateiname des neuen, umgestellten Realm lautet:

dbname.realmname.NEW

dbname

Name der Datenbank, die umgestellt wird

realmname

Name des Realm, der umgestellt wird.
Beim Database Directory steht *realmname* für DBDIR;
beim Database Compiler Realm steht *realmname* für DBCOM.

Speicherbedarf für den umgestellten Realm ermitteln

Anhaltspunkte, ob der umgestellte Realm mehr Platz beansprucht als der ursprüngliche Realm und wie groß somit die Primärzuweisung für den umgestellten Realm zu wählen ist, liefert die im Folgenden skizzierte Vorgehensweise von BPGSIZE bei der Umstellung.

Faktoren, die u.U. zusätzlichen Speicherplatz für den umgestellten Realm erfordern:

- Bei der Umstellung einer 2-Kbyte-Datenbank auf das 4-Kbyte- oder 8-Kbyte-Format wird die Seitenlänge von 2048 byte des ursprünglichen Realm nicht exakt verdoppelt bzw. vervierfacht. Da außerdem jeder Satz vollständig innerhalb *einer* Seite abgelegt werden muss, kann eine neue 4000 byte- bzw. 8096 byte-Seite in der Regel nicht ganz den Inhalt von zwei bzw. vier vollen 2048 byte-Seiten aufnehmen.
- Hashbereiche setzt BPGSIZE bei der Umstellung einer 2-Kbyte -Datenbank auf das 4-Kbyte- oder 8-Kbyte-Format so um, dass jede direkte oder indirekte CALC-Seite der Länge 2048 byte eins zu eins abgebildet wird auf eine neue CALC-Seite von 4000 byte bzw. 8096 byte Länge. Bei der Umstellung einer 4-Kbyte-Datenbank auf das 8-Kbyte-Format wird eine CALC-Seite der Länge 4000 byte eins zu eins auf eine neue CALC-Seite von 8096 byte Länge abgebildet. Somit benötigen Hashbereiche im umgestellten Realm zunächst den doppelten bzw. vierfachen Speicherplatz.
- Eine zu kleine TABLE-FILLING-Angabe bei der Anweisung CONVERT-DATABASE (siehe "CONVERT-DATABASE (Umstellung der Datenbank steuern)") bewirkt, dass BPGSIZE die Tabellen im neuen Realm nicht vollständig füllt. Falls sich dadurch der Füllgrad der Tabellen verringert, wird mehr Platz benötigt.

Faktoren, die u.U. den Speicherplatzbedarf für den umgestellten Realm verringern:

- Nur wenn eine Seite des ursprünglichen Realm bereits Sätze oder Tabellen enthält, übernimmt BPGSIZE diese Seite in eine Seite des umgestellten Realm; leere Seiten werden nicht berücksichtigt.
- BPGSIZE füllt die Seiten des umgestellten Realm so weit wie möglich mit Sätzen und Tabellen. Der Speicherplatzbedarf für den umgestellten Realm verringert sich, wenn der ursprüngliche Realm viele nur teilweise gefüllte Seiten enthält.
- BPGSIZE reduziert nach Möglichkeit die Anzahl der Überlaufseiten bei Hashbereichen und Duplikat-Tabellen.

Umgestellter Realm wird von BPGSIZE angelegt

Wird der Realm *dbname.realmname*.NEW von BPGSIZE angelegt, so legt BPGSIZE den FPA-Bereich von *dbname*. *realmname*.NEW in gleicher Größe an wie den FPA-Bereich des ursprünglichen Realm *dbname.realmname*.

Dabei ist zu beachten, dass der neue Realm dbname.realmname.NEW auf Grund von Dateierweiterungen zu groß werden kann für den von BPGSIZE ermittelten FPA-Bereich. In diesem Fall bricht BPGSIZE die Umstellung mit Fehlermeldung ab. Sie müssen dann mit dem BS2000-Kommando CREATE-FILE die Datei für den Realm *dbname. realmname.*NEW mit ausreichender Primärzuweisung selbst anlegen und die Umstellung des Realm mit BPGSIZE noch einmal neu starten.

Datei für den umgestellten Realm anlegen

Wenn der umgestellte Realm mehr Platz beansprucht als der ursprüngliche Realm, müssen Sie die Datei *dbname*. *realmname*.NEW für den umgestellten Realm mit dem BS2000-Kommando CREATE-FILE selbst anlegen (siehe auch Abschnitt „Maximalgröße für UDS/SQL-Dateien“, "Dateien und Realms einer UDS/SQL-Datenbank").

In diesem Fall errechnet BPGSIZE aus der bei CREATE-FILE gemachten Angabe zur Primärzuweisung die erforderliche Größe für den FPA-Bereich des umgestellten Realm. Sofern der ermittelte Wert größer ist als beim ursprünglichen Realm, legt BPGSIZE den FPA-Bereich für den umgestellten Realm in der errechneten Größe an. Falls Sie als Sekundärzuweisung einen Wert kleiner als 576 PAM-Seiten angeben, erhöht BPGSIZE die Sekundärzuweisung automatisch auf 576 PAM-Seiten.

Umstellung von Realms mit verteilbaren Listen

Bei der Umstellung von Realms, die an einer verteilbaren Liste beteiligt sind (und die daher in einem einzigen BPGSIZE Lauf umgesetzt werden müssen) müssen Sie beachten, dass die Umsetzung versucht, die Sätze der Membersatzart in der verteilbaren Liste auf die beteiligten Realms gleich zu verteilen.

Falls die Verteilung auf die Realms vorher sehr unterschiedlich war, sollten Sie die neuen Realms vor der Umstellung explizit unter Berücksichtigung der Gleichverteilung in entsprechender Größe anlegen (dazu können Sie sich mit BSTATUS über Lage und Verteilung der Sätze auf den verschiedenen Realms informieren). Wenn Sie die Realms nämlich nicht explizit anlegen, so werden sie in der aktuell vorhandenen Größe angelegt, ohne die angestrebte Gleichverteilung zu berücksichtigen. Dies könnte dann evtl. zu einem Abbruch bei der gleichverteilten Umsetzung führen, da BPGSIZE die neuen Realms nicht erweitert.

8.2.2.2 Benötigte Arbeitsdateien

Für die Aktualisierung der DBTTs der Datenbank benötigt BPGSIZE während der Umstellung eines Realm je eine Arbeitsdatei pro im Realm enthaltener Satzart.
Die Arbeitsdateien haben folgende Namen:

UTI.BPGSIZE.*dbname*.*realmref*.*recref*

dbname

Name der Datenbank, die umgestellt wird

realmref

dreistellige interne Nummer des Realm, der umgestellt wird

recref

fünfstellige interne Nummer der Satzart

Eine Arbeitsdatei mit *recref* 0 wird gegebenenfalls pro Realm für die Behandlung von SYSTEM-Sets benötigt. Eine Arbeitsdatei mit *realmref* 0 und *recref* 0 wird in der Abschlussbehandlung realmübergreifend benötigt.

Im Standardfall legt BPGSIZE während der Umstellung diese Dateien auf gemeinschaftlichem Datenträger an mit einer Primärzuweisung von ca.129 PAM-Seiten und einer Sekundärzuweisung von 25 PAM-Seiten.

Mit dem BS2000-Kommando CREATE-FILE können Sie Arbeitsdateien vor Ausführung von BPGSIZE auch gezielt unter einer bestimmten Katalogkennung oder auf privater Platte anlegen. Dabei müssen Sie stets den Standarddateinamen verwenden. Interne Realm- und Satzartnummer sowie die Zuordnung von Realm-Nummer und Satzartnummer ermitteln Sie mit dem Dienstprogramm BPSIA (siehe Handbuch „Sichern, Informieren und Reorganisieren“). Das Mengengerüst der zwischenzuspeichernden Daten ergibt sich für die satzspezifischen Arbeitsdateien aus der folgenden Formel

```
Anzahl der Sätze der Satzart im betreffenden Realm *
Anzahl DBTT Spalten der Satzart * 9 Bytes
```

Liegt die DBTT einer Satzart im selben Realm wie alle Sätze dieser Satzart, so verarbeitet BPGSIZE die zugehörige Arbeitsdatei gemeinsam mit dem Realm und löscht anschließend die Arbeitsdatei. Andernfalls löscht BPGSIZE die Arbeitsdatei erst, nachdem alle Realms (einschließlich DBDIR und DBCOM) sowie ggf. das COBOL Subschema Directory (COSSD, siehe nächster Abschnitt) erfolgreich umgestellt sind.

In der LMS-Bibliothek SIPLIB.UDS-SQL-T.024 wird die Beispielprozedur P.GEN-FILE-BPGSIZE bereitgestellt, die mittels CSV-Ausgabedaten der Dienstprogramme BPSIA und BSTATUS die satzspezifischen Arbeitsdateien erstellt.

8.2.2.3 COBOL Subschema Directory (COSSD) der umgestellten Datenbank

BPGSIZE erzeugt das umgestellte COBOL Subschema Directory (COSSD) nur dann, wenn bei der Umstellung die Seitenlänge der Datenbank vergrößert wird. In diesem Fall erzeugt BPGSIZE das umgestellte COSSD automatisch und legt es auf gemeinschaftlichem Datenträger in der Größe des ursprünglichen COSSD an.
Der Dateiname des umgestellten COSSD lautet:

dbname.COSSD.NEW

dbname

Name der Datenbank, die umgestellt wird

Mit dem BS2000-Kommando CREATE-FILE können Sie die Datei *dbname*.COSSD.NEW für das umgestellte COBOL Subschema Directory auch gezielt unter einer bestimmten Katalogkennung oder auf privater Platte anlegen.

Wenn die Seitenlänge der Datenbank bei der Umstellung mit BPGSIZE nicht verändert wird, ist das ursprüngliche COSSD inhaltlich auch für die umgestellte Datenbank gültig. Die Zuordnung zwischen COSSD und umgestellter Datenbank treffen Sie dann ggf. durch Kopieren oder Umkatalogisieren mit dem BS2000-Kommando COPY-FILE bzw. MODIFY-FILE-ATTRIBUTES.

8.2.2.4 Modulbibliothek für Hashroutinen (HASHLIB) der umgestellten Datenbank

Die Modulbibliothek für Hashroutinen (HASHLIB) wird von BPGSIZE bei der Datenbankumstellung nicht berücksichtigt, da die HASHLIB der ursprünglichen Datenbank inhaltlich auch für die umgestellte Datenbank gültig ist. Die Zuordnung zwischen HASHLIB und umgestellter Datenbank treffen Sie ggf. durch Kopieren oder Umkatalogisieren mit dem BS2000-Kommando COPY-FILE bzw. MODIFY-FILE-ATTRIBUTES.

8.2.3 Phasen der Umstellung

Wahlweise können Sie die gesamte Datenbank in *einem* BPGSIZE-Lauf umstellen oder die Umstellung der Datenbank auf mehrere BPGSIZE-Läufe verteilen.

Unabhängig davon, in wie vielen BPGSIZE-Läufen Sie die Datenbank umstellen, sind bei der Umstellung immer drei Phasen zu unterscheiden:

1. Umstellung des Database Directory (*dbname*.DBDIR)
2. Umstellung des Database Compiler Realm (*dbname*.DBCOM) und aller Benutzer-Realms (*dbname.realmname*)
3. Aktualisierungslauf: U.a. spielt BPGSIZE noch nicht verarbeitete Hilfsdateien ein und aktualisiert die restlichen DBTTs; bei Umstellung auf ein größeres Seitenformat wird auch das COBOL Subschema Directory (*dbname*.COSSD) umgestellt.

Den Ablaufmeldungen von BPGSIZE können Sie entnehmen, welcher Realm gerade bearbeitet wird.

Bei der Umstellung von Realms, die nicht an verteilbaren Listen beteiligt sind, werden jeweils nur der umzustellende Realm, sowie *dbname*.DBDIR und *dbname*.DBDIR.NEW benötigt. Daher können Sie alle anderen ursprünglichen und umgestellten Realms der Datenbank vorübergehend auslagern (um Platz für die Umstellung des gerade bearbeiteten Realms zu schaffen).

Einen Realm, der von einer verteilbaren Liste verwendet wird, können Sie nicht einzeln umstellen. Stattdessen müssen Sie alle Realms, die zu einer verteilbaren Liste *S* gehören, im gleichen BPGSIZE-Lauf umstellen. Die Menge der Realms, die in einem BPGSIZE-Lauf umgestellt werden kann, muss abgeschlossen gegenüber verteilbaren Listen sein.

Eine Menge von Realms ist dann abgeschlossen bezüglich verteilbarer Listen, wenn sie die Eigenschaft hat, dass für jeden Realm der Menge, der von einer beliebigen verteilbaren Liste *S* verwendet wird, auch jeder andere Realm, der von der gleichen Liste *S* verwendet wird, in der Menge liegt.
Ist beispielsweise *SX* eine verteilbare Liste mit den Realms *R1*, *R2*, *R3* und *SY* eine verteilbare Liste mit den Realms *R2*, *R4*, *R5*, so ist die Realmmenge *R1*, *R2*, *R3*, *R4*, *R5* abgeschlossen bezüglich verteilbarer Listen. Die Menge *R1*, *R2*, *R3* ist dagegen nicht abgeschlossen bezüglich verteilbarer Listen, da *R2* in der Menge liegt, nicht aber die zur gleichen Liste *SY* gehörigen Realms *R4* und *R5*.

Benötigte Realms

Die folgende Übersicht zeigt, welche Realms BPGSIZE für die Umstellung des Database Directory (*dbname*.DBDIR), des Database Compiler Realm (*dbname*.DBCOM) sowie der einzelnen Benutzerrealms benötigt. Die benötigten Realms müssen während der Dauer der Umstellung auf Magnetplatte vorliegen.

- Für die Umstellung des Database Directory werden benötigt:
  - ursprüngliches Database Directory *dbname*.DBDIR
  - Datei *dbname*.DBDIR.NEW für das umgestellte Database Directory (wird im Standardfall von BPGSIZE angelegt)
- Für die Umstellung des Database Compiler Realm werden benötigt:
  - ursprünglicher Database Compiler Realm *dbname*.DBCOM
  - Datei *dbname*.DBCOM.NEW für den umgestellten Database Compiler Realm (wird im Standardfall von BPGSIZE angelegt)
  - ursprüngliches Database Directory *dbname*.DBDIR
  - umgestelltes Database Directory *dbname*.DBDIR.NEW

- Für die Umstellung eines Benutzerrealm *dbname.realmname* werden benötigt:
  - ursprünglicher Benutzerrealm *dbname.realmname*
  - Datei *dbname.realmname*.NEW für den umgestellten Benutzerrealm (wird im Standardfall von BPGSIZE angelegt)
  - ursprüngliches Database Directory *dbname*.DBDIR
  - umgestelltes Database Directory *dbname*.DBDIR.NEW

Mit Ausnahme des ursprünglichen Database Directory (*dbname*.DBDIR) benötigt BPGSIZE einen ursprünglichen Realm nicht mehr für die Umstellung, sobald der betreffende Realm vollständig umgestellt ist. Das ursprüngliche Database Directory wird erst nach Umstellung der gesamten Datenbank nicht mehr benötigt.

Systemumgebung von BPGSIZE

Die folgenden Abbildungen zeigen die Systemumgebung von BPGSIZE in den einzelnen Phasen der Umstellung.

Systemumgebung bei der Umstellung von DBDIR

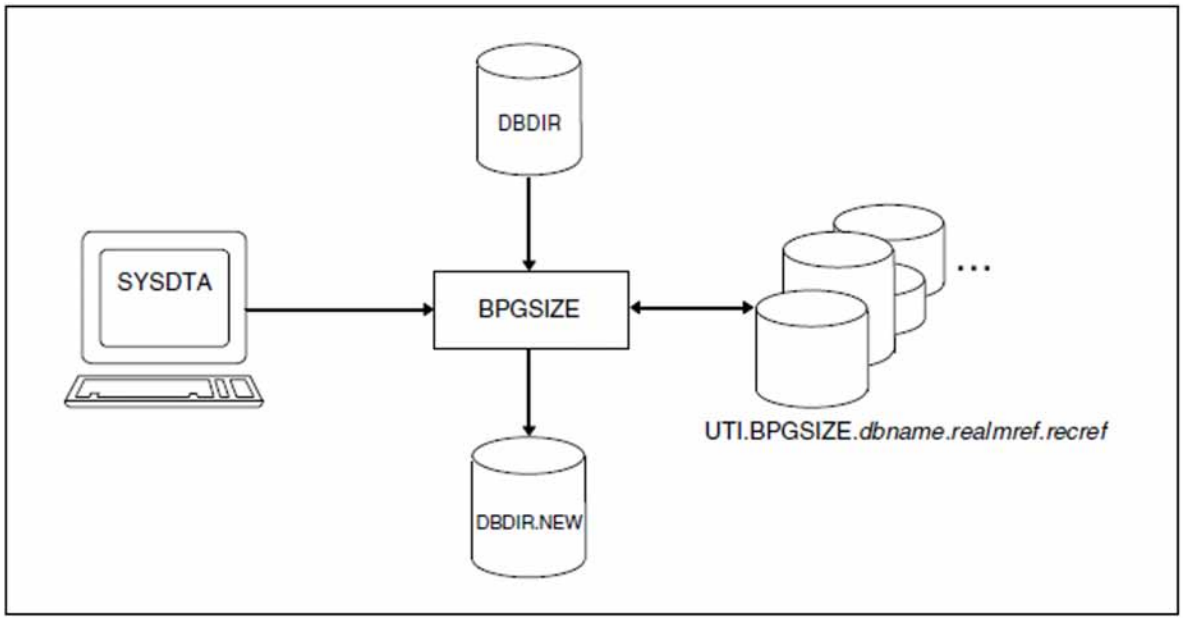

Bild 36: Systemumgebung von BPGSIZE bei der Umstellung des DBDIR

Systemumgebung bei der Umstellung von DBCOM und Benutzerrealms

Bei der Umstellung des Database Compiler Realm (*dbname*.DBCOM) oder eines Benutzerrealm (*dbname*.*realmname*) benötigt BPGSIZE folgende Systemumgebung:

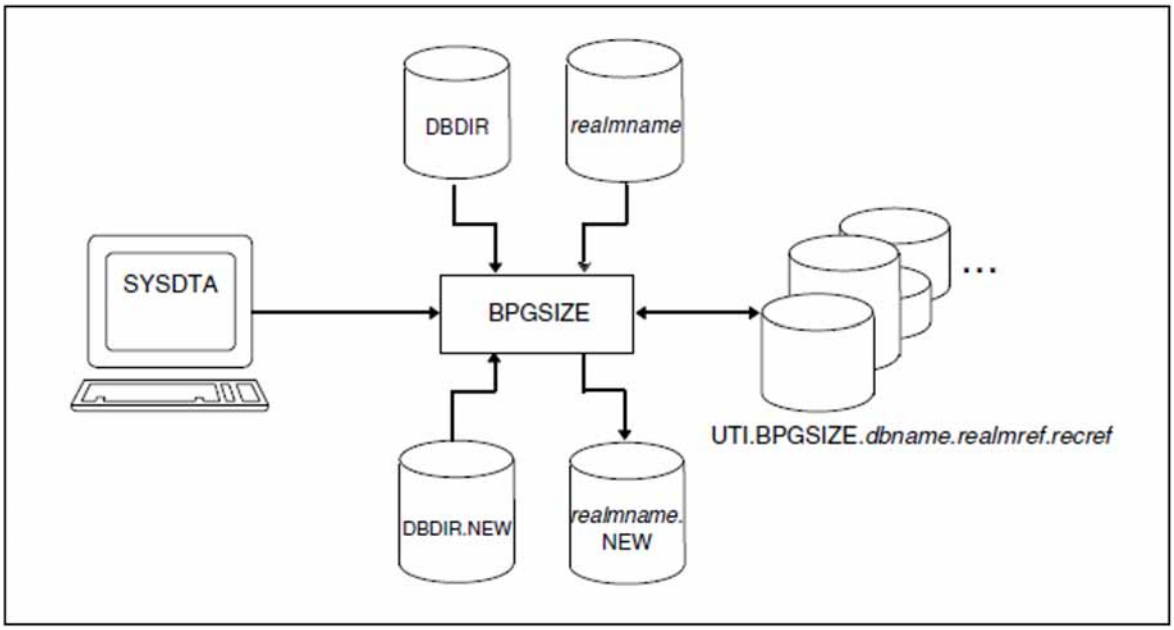

Bild 37: Systemumgebung von BPGSIZE bei der Umstellung des DBCOM oder eines Benutzerrealm

Im Fall des Database Compiler Realm steht in Bild 37 *realmname* für DBCOM.

Systemumgebung beim Aktualisierungslauf

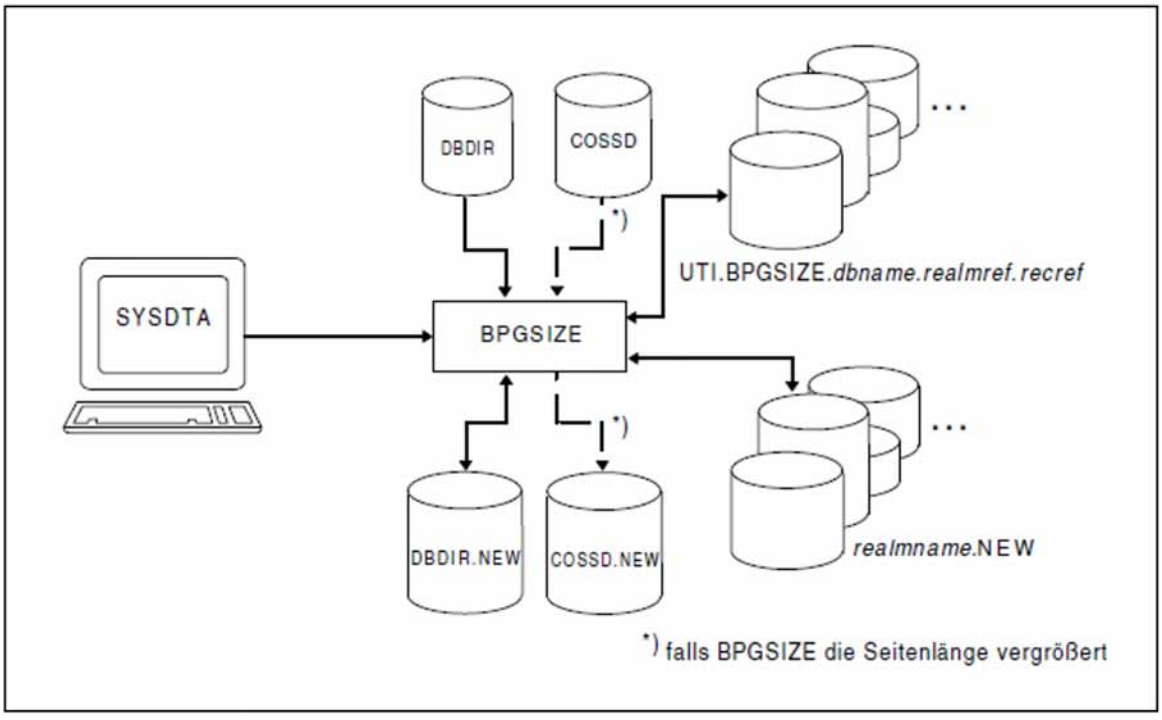

Bild 38: Systemumgebung von BPGSIZE beim Aktualisierungslauf

Das COBOL Subschema Directory (COSSD) wir nur umgestellt, wenn die Datenbank auf ein größeres Seitenformat umgestellt wird.

Datenbank in einem BPGSIZE-Lauf umstellen

Wenn Sie die gesamte Datenbank in einem einzigen BPGSIZE-Lauf umstellen, fasst BPGSIZE automatisch alle drei Phasen der Umstellung in diesem Lauf zusammen. Hierzu rufen Sie BPGSIZE z.B. mit der Angabe REALM-NAME=*ALL bei der CONVERT-DATABASE-Anweisung auf (siehe "CONVERT-DATABASE (Umstellung der Datenbank steuern)").
Bei der Datenbankumstellung in *einem* BPGSIZE-Lauf werden die einzelnen Realms nacheinander umgestellt. Dies kann bei größeren Datenbanken zu langen Umstellzeiten führen.
Deshalb empfiehlt es sich, nur kleine Datenbanken in *einem* BPGSIZE-Lauf umzustellen.

Datenbank in mehreren BPGSIZE-Läufen umstellen

Wenn Sie die Umstellung der Datenbank auf mehrere BPGSIZE-Läufe verteilen, müssen Sie selbst für die richtige Reihenfolge bei der Umstellung sorgen.

Beim Umstellen der Datenbank in mehreren BPGSIZE-Läufen gehen Sie wie folgt vor:

1. Mit dem ersten BPGSIZE-Lauf müssen Sie das umgestellte Database Directory (*dbname*.DBDIR.NEW) erzeugen.
   Zusätzlich können Sie bereits mit dem ersten BPGSIZE-Lauf auch den Database Compiler Realm (*dbname*.DBCOM) und/oder Benutzerrealms der Datenbank umstellen.
2. Anschließend stellen Sie die restlichen Realms der Datenbank in weiteren BPGSIZE-Läufen um. Dabei können Sie BPGSIZE für die einzelnen Realms auch parallel aufrufen. Dies kann sinnvoll sein, da die Dauer der Realmumstellung wesentlich von der Größe eines Realms abhängt.

   > **i** Ausnahme:
   > Da in einem BPGSIZE-Lauf nur Realmmengen umgestellt werden können, die bezüglich verteilbarer Listen abgeschlossen sind (siehe "Phasen der Umstellung"), können Realms einer verteilbaren Liste nicht parallel umgestellt werden.

   Generell ist bei der Entscheidung, Realms sequenziell oder parallel umzustellen, zu berücksichtigen:

   - Die sequenzielle Realmumstellung erfordert weniger Speicherplatz, führt aber im Allgemeinen zu längeren Laufzeiten.
   - Die parallele Realmumstellung erfordert kurzfristig mehr Speicherplatz, beschleunigt aber die Umstellung der Datenbank.
3. Sie starten den Aktualisierungslauf, indem Sie BPGSIZE mit der Angabe REALM-NAME=*ALL bei der CONVERT-DATABASE-Anweisung aufrufen (siehe "CONVERT-DATABASE (Umstellung der Datenbank steuern)"). Falls auf ein größeres Seitenformat umgestellt wird, erzeugt BPGSIZE dabei automatisch auch das umgestellte COBOL Subschema Directory (*dbname*.COSSD.NEW).
4. Alle Realms zu einer verteilbaren Liste müssen im gleichen BPGSIZE-Lauf umgesetzt werden.

Wiederanlauf beim Abbruch von BPGSIZE

Wird ein BPGSIZE-Lauf abgebrochen, z.B. wegen einem zu kleinen FPA-Bereich im umgestellten Realm, sollten Sie für den abgebrochenen BPGSIZE-Lauf einen Wiederanlauf durchführen. Für den Wiederanlauf starten Sie BPGSIZE erneut mit denselben Anweisungen. Auf diese Weise verhindern Sie, dass BPGSIZE auch bereits vollständig umgestellte Realms noch einmal bearbeitet.

8.2.4 Anweisungen für BPGSIZE

Für das Dienstprogramm BPGSIZE gibt es folgende Anweisungen:

Anweisung	Funktion
ALLOCATE-BUFFER-POOL	Puffergröße festlegen
CONVERT-DATABASE	Datenbank umstellen
END	Eingabe der Anweisungen beenden
OPEN-DATABASE	Datenbank eröffnen
UNDO	Anweisung rückgängig machen

Tabelle 49: Anweisungen für BPGSIZE

Die Anweisungen von BPGSIZE können Sie nicht nur auf die Benutzerrealms der Datenbank anwenden, sondern auch auf das Database Directory (DBDIR) und den Database Compiler Realm (DBCOM). Temporäre Realms dürfen Sie nicht angeben.

Regeln der Anweisungen

Falsch eingegebene Anweisungen können Sie korrigieren. Jede richtig eingegebene Anweisung können Sie zurücknehmen mit der Anweisung UNDO.

Widersprechen sich die Angaben bzgl. der Funktion bzw. des Objekts, so gilt immer die letzte Angabe.

BPGSIZE führt die Anweisungen ALLOCATE-BUFFER-POOL, OPEN-DATABASE und UNDO sofort aus. Alle gültigen CONVERT-DATABASE-Anweisungen führt BPGSIZE nach der Anweisung END aus.

Die Anweisungen müssen Sie in folgender Reihenfolge eingeben:

1. ALLOCATE-BUFFER-POOL
2. OPEN-DATABASE
3. CONVERT-DATABASE (ggf. mehrfach)
4. ggf. UNDO
5. END

Syntax der Anweisungen

Nachfolgend sind die einzelnen Anweisungen in alphabetischer Reihenfolge ausführlich beschrieben.

8.2.4.1 ALLOCATE-BUFFER-POOL (Puffergröße festlegen)

Mit der Anweisung ALLOCATE-BUFFER-POOL legen Sie die Größe des verwendeten Buffer Pools in Mbyte fest (siehe Handbuch „Datenbankbetrieb“).

Die Anweisung ALLOCATE-BUFFER-POOL müssen Sie als erste Anweisung angeben. Wenn Sie für die Pufferinitialisierung Standardwerte verwenden wollen, kann die Anweisung entfallen.

Die Anweisung ALLOCATE-BUFFER-POOL wird anschließend nicht mehr in der SDF-Maske angeboten.

Die Anweisung ALLOCATE-BUFFER-POOL kann durch die Anweisung UNDO nicht zurückgenommen werden.

ALLOCATE-BUFFER-POOL
BUFFER-SIZE = **STD** / <integer 1..2000>

BUFFER-SIZE = STD
Die Standard-Größe des Bufferpools wird mit 2 Mbyte festgelegt.

BUFFER-SIZE = <integer 1..2000>
Größe des Buffer Pools in Mbyte. Die Größe des Bufferpools muss innerhalb der angegebenen Grenzen liegen. Der Maximalwert ist abhängig von der Version des Betriebssystems, dem Hauptspeicherausbau der Anlage sowie dem kennungsspezifischen Wert von ADDRESS-SPACE-LIMIT.

8.2.4.2 CONVERT-DATABASE (Umstellung der Datenbank steuern)

Mit der Anweisung CONVERT-DATABASE steuern Sie die Umstellung der Realms.
BPGSIZE erzeugt dabei Realms der umgestellten Datenbank mit dem Kopienamen NEW. Bei jeder Datenbankumstellung muss als erster Realm das DBDIR umgestellt werden.

Die Anweisung CONVERT-DATABASE können Sie pro BPGSIZE-Lauf mehrfach angeben. Eine CONVERT-DATABASE-Anweisung ist erst dann gültig (d.h. kann erst dann ausgeführt werden), wenn alle in der Anweisung genannten Realms vorhanden sind. Wenn in der CONVERT-DATABASE-Anweisung ein bereits umgestellter Realm angegeben ist, wird dieser Realm *nicht* erneut umgestellt.

CONVERT-DATABASE

REALM-NAME = **<u>*ALL</u>** / ***ALL-EXCEPT(...)** / list-poss(30): <realmname>

***ALL-EXCEPT**(...)

| **NAME** = list-poss(30): <realmname>

,**DATABASE-PAGE-LENGTH** = **<u>*UNCHANGED</u>** / **2KB** / **4KB** / **8KB**

,**TABLE-FILLING** = ***UNCHANGED** / **<u>*MAXIMUM</u> /** <integer 1..100>

REALM-NAME = <u>*ALL</u>
Alle Realms der Datenbank, einschließlich DBDIR und DBCOM, werden umgestellt.
Bei jeder Datenbankumstellung müssen Sie einen BPGSIZE-Lauf mit nur einer CONVERT-DATABASE-Anweisung und der Angabe REALM-NAME=*ALL durchführen, und zwar

- als ersten und einzigen BPGSIZE-Lauf, wenn Sie die gesamte Datenbank auf einmal und mit gleichem Füllgrad (s.u.) für alle Tabellen umstellen möchten, oder
- als abschließenden Aktualisierungslauf, nachdem bereits alle Realms der Datenbank in vorausgegangenen BPGSIZE-Läufen umgestellt worden sind.

REALM-NAME = *ALL-EXCEPT(...)
Mit Ausnahme der angegebenen Realms werden alle Realms umgestellt.

NAME = list-poss(30): <realmname>
Namen der Realms, die nicht umgestellt werden. Hier können Sie auch die Realms DBDIR (Database Directory) und DBCOM (Database Compiler Realm) angeben.

REALM-NAME = list-poss(30): <realmname>
Alle angegebenen Realms werden umgestellt. Hier können Sie auch die Realms DBDIR (Database Directory) und DBCOM (Database Compiler Realm) angeben.

> In folgenden Fällen erzeugt BPGSIZE zu einem durch REALM-NAME=... spezifizierten Realm keinen umgestellten Realm:
>
> - Der Realm existiert nicht.
> - Der Realm ist abgeschaltet.
> - Der Realm ist inkonsistent.
> - Der Realm passt nicht zum ursprünglichen DBDIR (*dbname*.DBDIR) oder nicht zum umgestellten DBDIR (*dbname*.DBDIR.NEW).

DATABASE-PAGE-LENGTH = *UNCHANGED
Die Länge der Datenbankseiten bleibt bei der Umstellung unverändert. Falls im ursprünglichen Realm leere und nur teilweise gefüllte Seiten vorkommen, können Sie mit dieser Angabe den Speicherplatzbedarf des Realm verkleinern.

DATABASE-PAGE-LENGTH = 2KB
Die Länge der Datenbankseiten in der umgestellten Datenbank beträgt 2048 byte. Diese Angabe ist nur zulässig bei 2-Kbyte-Datenbanken und hat dort dieselbe Wirkung wie die Angabe *UNCHANGED. Bei einer 4-Kbyte- oder 8-Kbyte-Datenbank wird die Anweisung abgewiesen.

DATABASE-PAGE-LENGTH = 4KB
Die Länge der Datenbankseiten in der umgestellten Datenbank beträgt 4000 byte. Diese Angabe ist nur zulässig bei 2-Kbyte- und 4-Kbyte-Datenbanken und hat bei einer 4-Kbyte-Datenbank dieselbe Wirkung wie die Angabe *UNCHANGED. Bei einer 8-Kbyte-Datenbank wird die Anweisung abgewiesen.

DATABASE-PAGE-LENGTH = 8KB
Die Seitenlänge der umgestellten Datenbank beträgt 8096 byte. Diese Angabe hat bei einer 8-Kbyte-Datenbank dieselbe Wirkung wie die Angabe *UNCHANGED.

i Die Seitenlänge in der umgestellten Datenbank muss einheitlich sein. Deshalb ist bei den Angaben zu DATABASE-PAGE-LENGTH Folgendes zu beachten:

- Beim ersten BPGSIZE-Lauf für die Datenbank:
  Bei mehreren CONVERT-DATABASE-Anweisungen mit unterschiedlichen DATABASE-PAGE-LENGTH-Angaben berücksichtigt BPGSIZE stets die DATABASE-PAGE-LENGTH-Angabe der letzten gültigen CONVERT-DATABASE-Anweisung. Diese Angabe bestimmt somit die Seitenlänge in der umgestellten Datenbank. Eine Ablaufmeldung von BPGSIZE informiert Sie darüber.
- Bei allen weiteren BPGSIZE-Läufen für die Datenbank:
  Sie müssen bei allen DATABASE-PAGE-LENGTH-Angaben die im ersten BPGSIZE-Lauf festgelegte Seitenlänge spezifizieren. Andernfalls bricht BPGSIZE den Lauf mit Fehlermeldung ab.

TABLE-FILLING = *UNCHANGED
Der Füllgrad der Tabellen in den umgestellten Realms bleibt (nahezu) unverändert.

TABLE-FILLING = *MAXIMUM
In den umgestellten Realms wird jede Tabelle wie folgt gefüllt:

- Auf Stufe 0 (Grundstufe) bleibt ein Tabelleneintrag frei.
- Auf Stufe 1 wird die Tabelle zu 95 % gefüllt.
- Auf allen darüberliegenden Stufen bleibt jeweils ein Tabelleneintrag frei.

TABLE-FILLING = <integer 1..100>
In den umgestellten Realms wird jede Tabelle wie folgt gefüllt:

- Auf Stufe 0 (Grundstufe) wird die Tabelle gemäß dem angegebenen Füllgrad (%) gefüllt.
- Auf Stufe 1 wird die Tabelle zu 95 % gefüllt.
- Auf allen darüberliegenden Stufen bleibt jeweils ein Tabelleneintrag frei.

Wird in mehreren CONVERT-DATABASE-Anweisungen mit unterschiedlichen TABLE-FILLING-Angaben derselbe Realm angesprochen, so gilt für den zugehörigen umgestellten Realm die TABLE-FILLING-Angabe der letzten gültigen CONVERT-DATABASE-Anweisung, in der dieser Realm angesprochen wird.

Da alle Tabellen eines umgestellten Realm mit dem angegebenen Prozentsatz gefüllt werden, benötigt ein umgestellter Realm u.U. mehr Speicherplatz als der zugehörige ursprüngliche Realm. Im Allgemeinen empfiehlt sich daher ein hoher Füllgrad (d.h. 99, 100 oder *MAXIMUM). Nähere Einzelheiten zur Bestimmung eines sinnvollen Füllgrads finden Sie im Kapitel „SSL“ des Handbuchs „Entwerfen und Definieren“.

8.2.4.3 END (Eingabe der Anweisungen beenden)

Mit der Anweisung END beenden Sie die Eingabe der Anweisungen. Die Ausführung wird gestartet.

END

Diese Anweisung hat keine Operanden.

8.2.4.4 OPEN-DATABASE (Datenbank eröffnen)

Mit der Anweisung OPEN-DATABASE legen Sie die Datenbank fest, die mit den nachfolgenden Anweisungen bearbeitet werden soll.

Die Anweisung OPEN-DATABASE ist unzulässig, wenn die Datenbank über SET-FILE-LINK LINK-NAME=DATABASE zugewiesen ist.

OPEN-DATABASE
DATABASE-NAME = <dbname> ,**COPY-NAME** = ***NONE** / <copyname> ,**USER-IDENTIFICATION** = ***OWN** / <userid>

DATABASE-NAME = <dbname>
Name der Datenbank. Sie können nur eine Datenbank bearbeiten, die in Ihrer eigenen Kennung liegt. Eine Datenbank aus einer fremden Kennung kann nur von der TSOS-Kennung des Systemverwalters bearbeitet werden.

COPY-NAME = *NONE
Es wird das Datenbank-Original bearbeitet.

COPY-NAME = <copyname>
Es wird die Schattendatenbank mit dem angegebenen Kopienamen bearbeitet.

USER-IDENTIFICATION = *OWN
Die Datenbank liegt in der eigenen Kennung.

USER-IDENTIFICATION = <userid>
Die Angabe einer fremden Datenbank-Kennung ist nur unter der TSOS-Kennung erlaubt.

8.2.4.5 UNDO (Anweisung rückgängig machen)

Die zuletzt eingegebene, korrekte Anweisung (außer UNDO selbst) wird nicht ausgeführt.

Die Anweisung UNDO setzt keine Anweisung ALLOCATE-BUFFER-POOL zurück.

Durch eine Folge von zwei UNDO-Anweisungen werden die beiden unmittelbar vor der UNDO-Folge stehenden Anweisungen (außer UNDO) nicht ausgeführt, durch eine Folge von drei UNDO-Anweisungen werden die drei unmittelbar vor der UNDO-Folge stehenden Anweisungen (außer UNDO) nicht ausgeführt usw.

UNDO

Diese Anweisung hat keine Operanden.

8.2.5 Kommandofolge zum Starten von BPGSIZE

Die hier beschriebene Kommandofolge geht davon aus, dass UDS/SQL mit IMON installiert wurde (siehe Abschnitt „START-Kommandos der UDS/SQL-Programme").

```
[01] /ADD-FILE-LINK LINK-NAME=DATABASE
       ,FILE-NAME=[:catid:][$userid.].DBDIR.[copyname]

[02] /CREATE-FILE FILE-NAME=[:catid:][$userid.]dbname.realmname.NEW
      [,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primär
       ,SECONDARY-ALLOCATION=576)) /
       ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn -
        DEVICE-TYPE=gerät[,SPACE=...])]

[03] ... weitere CREATE-FILE-Anweisungen für Dateien der umgestellten Realms

[04] /CREATE-FILE FILE-NAME= -
    [:catid:][$userid.]UTI.BPGSIZE.dbname.realmnummer.satzartnummer
      [,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primär
       ,SECONDARY-ALLOCATION=sekundär)) /
       ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn -
        DEVICE-TYPE=gerät[,SPACE=...])]

[05] ... weitere CREATE-FILE-Anweisungen für Arbeitsdateien von BPGSIZE

[06] /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK

[07] /START-UDS-BPGSIZE

[08] //OPEN-DATABASE DATABASE-NAME = ...

[09] //BPGSIZE-Anweisungen

[10] //END
```

01,08 Sie müssen genau eine der beiden Anweisungen angeben.

02 Mit diesem CREATE-FILE-Kommando können Sie wahlweise eine Datei für den umgestellten Realm (DBDIR, DBCOM oder Benutzerrealm) einrichten (siehe "Datei für den umgestellten Realm anlegen", "Realms der umgestellten Datenbank").

04 Mit diesem CREATE-FILE-Kommando können Sie wahlweise eine Arbeitsdatei für BPGSIZE einrichten (siehe "Benötigte Arbeitsdateien").

06 Die angegebene Version von BPGSIZE wird ausgewählt.
Die Angabe der Version wird generell empfohlen, da mehrere UDS/SQL-Versionen parallel installiert sein können.

07 Das UDS/SQL-Dienstprogramm kann auch mit den Aliasnamen BPGSIZE und START-UDS-PAGE-RESIZING gestartet werden.

8.2.6 Beispiel zu BPGSIZE

Für das folgende Beispiel wird angenommen, dass die Datenbank REISEN zunächst im 2-Kbyte-Format vorliegt. Das Beispiel zeigt die Umstellung dieser 2-Kbyte-Datenbank auf das 4-Kbyte-Format. Da es sich um eine kleine Datenbank handelt, ist es sinnvoll, die Datenbank in einem einzigen BPGSIZE-Lauf umzustellen. Die Arbeitsdateien werden von BPGSIZE angelegt.

```
/CREATE-FILE FILE-NAME=REISEN.DBDIR.NEW,SUPPORT=PUBLIC-DISK(SPACE=RELATIVE -
/   (PRIMARY-ALLOCATION=200,SECONDARY-ALLOCATION=50))
/CREATE-FILE FILE-NAME=REISEN.DBCOM.NEW,SUPPORT=PUBLIC-DISK(SPACE=RELATIVE -
/   (PRIMARY-ALLOCATION=550,SECONDARY-ALLOCATION=50))
/CREATE-FILE FILE-NAME=REISEN.REISENRLM.NEW,SUPPORT=PUBLIC-DISK(SPACE=RELATIVE -
/   (PRIMARY-ALLOCATION=250,SECONDARY-ALLOCATION=50))
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=REISEN.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.9B00
/START-UDS-BPGSIZE
```

```
***** START       BPGSIZE     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:11
```

```
//CONVERT-DATABASE REALM-NAME=*ALL,DATABASE-PAGE-LENGTH=4KB
//END
```

```
***** BEGIN        FUNCTION CONVERT DATABASE AT 2017-06-28 11:26:11
***** CONVERSION OF REALM DBDIR STARTED
                  CALC FOR RECORD USERGROUP-RECORD CONVERTED
                  CALC FOR RECORD SUBSCHEMA-RECORD CONVERTED
                  CALC FOR RECORD ERROR-MESSAGE CONVERTED
***** CONVERSION OF REALM DBDIR FINISHED
***** CONVERSION OF REALM DBCOM STARTED
***** CONVERSION OF REALM DBCOM FINISHED
***** CONVERSION OF REALM REISENRLM STARTED
                  CALC FOR RECORD VERKEHRSMITTEL CONVERTED
                  CALC FOR RECORD ARRANGEMENT CONVERTED
                  CALC FOR RECORD HOTEL CONVERTED
***** CONVERSION OF REALM REISENRLM FINISHED
***** NORMAL   END FUNCTION CONVERT DATABASE AT 2017-06-28 11:26:12

***** DIAGNOSTIC SUMMARY OF BPGSIZE

             NO WARNINGS
             NO ERRORS
             NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES  :         345
***** NORMAL END   BPGSIZE     (UDS/SQL  V2.9  0000 )     2017-06-28   11:26:12
```

8.3 Umgestellte Datenbank für den DB-Betrieb vorbereiten

Mit den nachfolgend beschriebenen Tätigkeiten erstellen Sie aus der mit BPGSIZE erzeugten Datenbankkopie (Kopiename NEW) eine einsatzfähige UDS/SQL-Datenbank:

1. umgestellte Datenbank zur Originaldatenbank erklären
2. evtl. Konsistenz der Datenbank prüfen und Datenbank-Informationen ausgeben
3. umgestellte Datenbank reorganisieren und anpassen:
   - Hashbereiche der umgestellten Datenbank reorganisieren
   - Größe der Realms in der umgestellten Datenbank anpassen (ggf.)
   - Probable Position Pointer (PPP) aktualisieren
   - Satzmengengerüst erweitern für Satzarten, zu denen künftig mehr als 16 777 215 Sätze abgespeichert werden sollen.
4. evtl. Konsistenz der Datenbank prüfen und Datenbank-Informationen ausgeben
5. ggf. AFIM-Logging wieder einschalten bzw. Online-Sicherungsfähigkeit wieder einstellen, Sicherungskopie der einsatzfähigen Datenbank erstellen

Für die genannten Tätigkeiten benötigen Sie u.a. die Dienstprogramme BMEND, BCHECK, BPSIA, BSTATUS, BPRECORD und BREORG, die im Handbuch „Sichern, Informieren und Reorganisieren“ beschrieben sind. Außerdem benötigen Sie evtl. die Dienstprogramme BCHANGE und BALTER, die im Kapitel „Datenbank umstrukturieren (BCHANGE, BALTER)" des vorliegenden Handbuchs beschrieben sind.

Umgestellte Datenbank zur Originaldatenbank erklären

Nach der Umstellung mit BPGSIZE erklären Sie die von BPGSIZE erzeugte Datenbankkopie (Kopiename NEW) zur Originaldatenbank. Hierzu verwenden Sie wahlweise das BS2000-Kommando MODIFY-FILE-ATTRIBUTES (Datenbank umkatalogisieren) oder das BS2000-Kommando COPY-FILE (Datenbank kopieren).

Falls die Realms der Datenbank online erweiterbar sein sollen, können Sie jetzt die Voraussetzungen dafür schaffen: Ein Realm, der online erweiterbar sein soll, muss zum Zeitpunkt der Erweiterung über genügend Speicherplatz verfügen. Diese Bedingung ist erfüllt, wenn der Realm zum Zeitpunkt der Erweiterung entweder über eine ausreichende Anzahl von Dateiseiten verfügt oder wenn für den Realm die Möglichkeit der Online-Realm-Erweiterung gegeben ist (siehe Handbuch „Datenbankbetrieb“, Online-Realm-Erweiterung). Die Zuweisung von Speicherplatz erfolgt beim Anlegen der Realm-Datei mit dem BS2000-Kommando CREATE-FILE (siehe Abschnitt „Datenbank-Aufbau vorbereiten"). Mit dem Kommando MODIFY-FILE-ATTRIBUTES können Sie die Speicherplatzzuweisung nachträglich ändern.

Sie können nun die umgestellte Datenbank bearbeiten.

Konsistenz der Datenbank prüfen und Datenbank-Informationen ausgeben

Bevor Sie mit der Reorganisation und Anpassung der umgestellten Datenbank beginnen, können Sie mit dem Dienstprogramm BCHECK die Konsistenz der Datenbank prüfen. Dabei sind jedoch die Laufzeiten des Dienstprogramms BCHECK zu berücksichtigen.
Mit den Dienstprogrammen BPSIA, BSTATUS und BPRECORD können Sie sich Datenbank-Informationen ausgeben lassen, um Anhaltspunkte für erforderliche Reorganisations- und Anpassungsmaßnahmen zu erhalten.

Nach Abschluss der Reorganisations- und Anpassungsmaßnahmen können Sie den BCHECK-Lauf sowie ggf. die BPSIA-, BSTATUS-, und BPRECORD-Läufe wiederholen.
Wegen der u.U. langen Laufzeit können Sie den BCHECK-Lauf auch entkoppelt von den Maßnahmen zum Anpassen der Datenbank zu einem späteren Zeitpunkt nachholen.

Umgestellte Datenbank reorganisieren und anpassen

Hashbereiche der umgestellten Datenbank reorganisieren

BPGSIZE füllt die Seiten der umgestellten Realms so weit wie möglich mit Sätzen und Tabellen und übernimmt keine leeren Seiten in die umgestellten Realms. Somit enthält ein umgestellter Realm zunächst keinen freien Platz für die Aufnahme neuer Daten. Andererseits bildet BPGSIZE beim Umstellen der Hashbereiche jede direkte und indirekte CALC-Seite eines ursprünglichen Realm eins zu eins ab auf eine ca. doppelt bzw. vierfach so große CALC-Seite im umgestellten Realm (falls auf ein größeres Seitenformat umgestellt wird).

Mit dem Dienstprogramm BREORG (Anweisung REORGANIZE-CALC) können Sie den Speicherplatzbedarf für Hashbereiche im umgestellten Realm verkleinern und so Platz für die Aufnahme neuer Daten schaffen.

Beispiel

```
/REMARK **** HASH-BEREICHE REORGANISIEREN ****
/START-UDS-BREORG
//SPECIFY-SCHEMA SCHEMA-NAME=schemaname
//REORGANIZE-CALC RECORD-NAME=satzname-1, -
                  CALC-RECORD=*WITHIN-POPULATION(REALM=*ALL, -
                                 POPULATION=1000), -
                  CALC-SEARCHKEY=*KEY-POPULATION(KEY-REF=*ALL, -
                                 POPULATION=1000)
//REORGANIZE-CALC RECORD-NAME=satzname-2, -
                  CALC-RECORD=*WITHIN-POPULATION(REALM=*ALL, -
                                 POPULATION=15000), -
                  CALC-SEARCHKEY=*KEY-POPULATION(KEY-REF=*ALL, -
                                 POPULATION=150000)
.
.
.
//END
```

Größe der umgestellten Realms anpassen

Reicht der durch die Reorganisation der Hashbereiche gewonnene zusätzliche Platz nicht aus, so müssen Sie die betreffenden Realms mit der Anweisung MODIFY-REALM-SIZE des Dienstprogramms BREORG vergrößern.

Wenn in einem umgestellten Realm zum Einspeichern neuer Daten weniger Platz benöigt wird, als bereits vorhanden ist, können Sie mit MODIFY-REALM-SIZE den Realm nach der Reorganisation der Hashbereiche ggf. auch verkleinern.

Bei der Anpassung der Realm-Größe sind insbesondere auch die Realms DBDIR (Database Directory) und DBCOM (Database Compiler Realm) zu berücksichtigen.

Probable Position Pointer (PPP) aktualisieren

Adressverweise, die als Probable Position Pointer (PPP) angelegt sind, werden von BPGSIZE beim Umstellen der Datenbank nicht aktualisiert. Nach der Umstellung können Sie sich anhand des SIA-Protokolls des Dienstprogramms BPSIA über falsch zeigende Probable Position Pointer (PPP) informieren (siehe PPP-BITS bei der Ausgabe von CALC/KEY/CALC-SEARCH-KEY INFORMATION) und diese gezielt für jeden Zugriffsweg mit der Anweisung REORGANIZE-SET des Dienstprogramms BREORG korrigieren. Dabei müssen Sie Sets, bei denen Sätze verlagert werden, zuerst reorganisieren. Solche Sets sind in der SSL mit MODE IS LIST definiert (siehe Handbuch „Entwerfen und Definieren“).

Zu beachten ist außerdem die lange Laufzeit der BREORG-Läufe zum Aktualisieren der Probable Position Pointer. Sie können die BREORG-Läufe zum Aktualisieren der Probable Position Pointer auch zeitlich von Vorbereitung der Datenbank für den DB-Betrieb entkoppeln und zu einem späteren Zeitpunkt nachholen.

Beispiel

```
/REMARK **** PPPs aktualisieren ****
/START-UDS-BREORG
//SPECIFY-SCHEMA SCHEMA-NAME=schemaname
//REORGANIZE-SET SET-NAME=setname-1,OWNER-SELECTION=ALL,FILLING=UNCHANGED
//REORGANIZE-SET SET-NAME=setname-2,OWNER-SELECTION=ALL,FILLING=UNCHANGED
.
.
.
//REORGANIZE-SET SET-NAME=setname-n,OWNER-SELECTION=ALL,FILLING=UNCHANGED
//END
```

Schneller verläuft die Aktualisierung der Probable Position Pointer mit der Anweisung REORGANIZE-POINTERS (Beschreibung siehe Kapitel „Datenbank reorganisieren mit BREORG“ im Handbuch „Sichern, Informieren und Reorganisieren“).

```
/START-UDS-BREORG
//SPECIFY-SCHEMA SCHEMA-NAME=schemaname
//REORGANIZE-POINTERS REALM-NAME=realmname-1
//REORGANIZE-POINTERS REALM-NAME=realmname-2
.
.
.
//REORGANIZE-POINTERS REALM-NAME=realmname-n,
//END
```

Beim Aktualisieren der Probable Position Pointer (PPP) von DBDIR und DBCOM verwenden sie folgende Schema- und Setnamen:

	Schemaname	**Setname**
DBDIR	PRIVACY-AND-IQF-SCHEMA	USERGROUP-USERID
DBCOM	COMPILER-SCHEMA	NAME-TABLE

Beispiel

```
/REMARK **** REORGANISATION VON DBDIR ****
/START-UDS-BREORG
//SPECIFY-SCHEMA SCHEMA-NAME=PRIVACY-AND-IQF-SCHEMA
//REORGANIZE-SET SET-NAME=USERGROUP-USERID,OWNER-SELECTION=ALL
//END
/REMARK **** REORGANISATION VON DBCOM ****
/START-UDS-BREORG
//SPECIFY-SCHEMA SCHEMA-NAME=COMPILER-SCHEMA
//REORGANIZE-SET SET-NAME=NAME-TABLE,OWNER-SELECTION=ALL
//END
```

Satzmengengerüst erweitern

Wenn die umgestellte Datenbank aus einer ursprünglichen 2-Kbyte-Datenbank hervorgegangen ist, müssen Sie das Satzmengengerüst für Satzarten erhöhen, zu denen künftig mehr als die in 2-Kbyte-Datenbanken maximal zulässigen 16 777 215 Sätze abgespeichert werden sollen.
Das Satzmengengerüst erweitern Sie mit der Anweisung MODIFY-RECORD-POPULATION des Dienstprogramms BREORG.

AFIM-Logging und Online-Sicherungsfähigkeit einschalten, einsatzfähige Datenbank sichern

Wenn Sie die umgestellte Datenbank mit AFIM-Logging betreiben wollen, schalten Sie mit dem Dienstprogramm BMEND das AFIM-Logging für die umgestellte Datenbank ein.

Wenn Sie die Möglichkeit haben wollen, die umgestellte Datenbank zukünftig online sichern zu können, schalten Sie mit dem Dienstprogramm BMEND die Online-Sicherungsfähigkeit ein.

Vor Wiederaufnahme des Datenbankbetriebs sollten sie unbedingt eine konsistente Sicherungskopie der umgestellten Datenbank erstellen.

8.4 Umgestellte Datenbank umstrukturieren

In einer vom 2-Kbyte-Format auf das 4-Kbyte- bzw. 8-Kbyte-Format umgestellten Datenbank können Sie den erweiterten Wertebereich für die maximale Anzahl Sätze pro Satzart ohne Änderung des Datenbankschemas nutzen, es sei denn, in der Schema-DDL ist innerhalb der Satzbeschreibung oder der LOCATION-MODE-Klausel ein Feld vom Typ DATABASE-KEY definiert - dieses müssen sie zur Nutzung des erweiterten Wertebereichs für Sätze pro Satzart auf DATABASE-KEY-LONG umstellen.
Dies hat den Vorteil, dass Anwenderprogramme zeitlich von der Datenbankumstellung entkoppelt angepasst werden können.
Änderungen im Schema sind erforderlich, wenn Sie für Satzarten mit explizit definiertem Database-Key-Feld den erweiterten Wertebereich uneingeschränkt nutzen wollen.

In diesem Abschnitt sind nur die zur Nutzung des erweiterten Wertebereichs erforderlichen Änderungen von Schema-DDL und Subschema-DDL dargestellt. Die Umstrukturierung einer Datenbank ist ausführlich beschrieben im Kapitel „Datenbank umstrukturieren (BCHANGE, BALTER)".

Wenn DB-Anwenderprogramme Database-Key-Werte einer Satzart verwenden, zu der auch künftig nicht mehr als 16 777 215 Sätze gespeichert werden (RSQ < 2^{24}-1), müssen Sie beim Umstrukturieren der Datenbank darauf achten, dass die betreffende Satzart eine Satzartnummer (REC-REF) < 255 erhält. Auf diese Weise lässt sich der Aufwand für Anpassung und ggf. Neuübersetzung der Anwenderprogramme reduzieren, da dann entsprechende USAGE IS DATABASE-KEY-Felder nicht angepasst werden müssen (siehe Abschnitt „COBOL- und CALL-DML-Anwendungen anpassen").

Bereits vorhandene Database-Key-Werte werden in das erweiterte Format umgesetzt, wenn Sie bei der Umstrukturierung der Datenbank den Datenbestand mit dem Dienstprogramm BALTER anpassen (siehe Abschnitt „Schemaänderungen analysieren und Datenbestand anpassen mit BALTER").

Um Ihre Datenbank zu optimieren, sollten Sie die Angaben zum Mengengerüst in der SSL an die erweiterten Wertebereiche anpassen (siehe Handbuch „Entwerfen und Definieren").

Datenbankschema anpassen

Folgende Sprachmittel der Schema-DDL müssen Sie für die Nutzung erweiterter Database-Key-Werte anpassen:

- TYPE IS DATABASE-KEY ändern in TYPE IS DATABASE-KEY-LONG
- LOCATION MODE IS DIRECT ändern in LOCATION MODE IS DIRECT-LONG

Bei Satzarten, die ein Database-Key-Feld (TYPE IS DATABASE-KEY) enthalten, prüfen Sie, ob dieses Database-Key-Feld künftig erweiterte Database-Key-Werte aufnehmen soll. Dabei ist zu beachten, dass ein Database-Key-Feld nicht in jedem Fall für den Database Key des Satzes verwendet werden muss, in dem es definiert ist. Es kann z.B auch für den Database Key einer anderen Satzart verwendet werden. Für die Entscheidung, ob ein Feld vom Typ DATABASE-KEY in ein DATABASE-KEY-LONG-Feld zu ändern ist, sollten Sie deshalb auch mögliche Querverbindungen untersuchen.
Die Änderung eines DATABASE-KEY-Feldes in ein DATABASE-KEY-LONG-Feld erfordert außerdem die Änderung aller LOCATION MODE IS DIRECT-Klauseln in LOCATION MODE IS DIRECT-LONG, in denen dieses DATABASE-KEY-LONG-Feld angesprochen wird (siehe Handbuch „Entwerfen und Definieren").

Subschemata anpassen

Zu jedem in den Typ DATABASE-KEY-LONG geänderten Database-Key-Feld des Datenbankschemas müssen Sie in den betroffenen Subschema-Beschreibungen die zugehörigen USAGE IS DATABASE-KEY-Klauseln ändern in USAGE IS DATABASE-KEY-LONG (siehe Handbuch „Entwerfen und Definieren"). Anschließend müssen Sie diese Subschemata sowie die darauf zugreifenden Anwenderprogramme neu übersetzen.

Wenn alle Satzarten eines Subschemas, die ein DATABASE-KEY-LONG-Feld enthalten, mit der Subschema-DDL-Anweisung COPY übernommen werden, müssen Sie dieses Subschema nicht ändern, sondern nur neu übersetzen.

Welche Subschemata angepasst werden müssen, ermitteln Sie nach dem BALTER-Lauf zur Datenbestandsanpassung (siehe "Schemaänderungen analysieren und Datenbestand anpassen mit BALTER") mit dem DDL-Compiler-Lauf zum Übernehmen der Subschemata, der inkompatible Subschemata herausfiltert (siehe Abschnitt „Kompatible Subschemata übernehmen").

8.5 COBOL- und CALL-DML-Anwendungen anpassen

Die Umstellung einer 2-Kbyte-Datenbank mit dem Dienstprogramm BPGSIZE auf das 4-Kbyte- oder 8-Kbyte-Format hat keine Auswirkungen auf angeschlossene Datenbank-Anwendungen und andere Datenbanken, solange die Datenbank ausschließlich Database-Key-Werte mit einer REC-REF <= 254 und einer RSQ <=2^{24}-1 enthält.

In den meisten Fällen wird man jedoch eine Datenbank umstellen, um in der Datenbank den erweiterten Wertebereich für Database-Key-Werte (REC REF > 254 und/oder
RSQ > 2^{24}-1) zu nutzen. Die Nutzung erweiterter Database-Key-Werte hat in der Regel Auswirkungen auf zugehörige Anwenderprogramme sowie u.U. auch auf andere Datenbanken und erfordert entsprechende Anpassungsmaßnahmen.

In folgenden Fällen kann es notwendig sein, ein Anwenderprogramm für die Nutzung erweiterter Database-Key-Werte in der Datenbank anzupassen:

- Die Schema-DDL der umgestellten Datenbank enthält mindestens eine der folgenden Klauseln:
  - TYPE IS DATABASE-KEY-LONG
  - LOCATION MODE IS DIRECT-LONG
  - SET SELECTION THRU LOCATION MODE OF OWNER, falls der Primärschlüssel mit LOCATION MODE IS DIRECT-LONG vereinbart ist
- Das Anwenderprogramm verwendet DML-Anweisungen, die über Database-Key-Felder auf die Datenbank zugreifen.
- Das Anwenderprogramm verwendet mindestens eine der folgenden COBOL-Definitionen:
  - Definition eines COBOL-Feldes vom Datentyp USAGE IS DATABASE-KEY
  - Redefinition mit REDEFINES für ein USAGE IS DATABASE-KEY-Feld
  - Verweis über eine LINKAGE auf eine Datenstruktur, die Felder vom Typ USAGE IS DATABASE-KEY-LONG enthält (z.B. LINKAGE auf ein Subschema).
- Das Anwenderprogramm verwendet Database-Key-Werte, ohne dass dies unmittelbar erkennbar ist (siehe Abschnitt „Weitere Stellen im Anwenderprogramm anpassen").

Nach der Anpassung muss das Anwenderprogramm neu übersetzt und gebunden werden.

8.5.1 DDL-Klauseln, die auf Verwendung erweiterter Database-Key-Werte hinweisen

Die Analyse von Schema- und Subschema-DDL liefert Hinweise, ob und ggf. wo in einem Anwenderprogramm Database-Key-Werte des erweiterten Wertebereichs verwendet werden.

Folgende DDL-Klauseln lassen auf die Verwendung erweiterter Database-Key-Werte im Anwenderprogramm schließen:

- Definition eines Feldes mit TYPE IS DATABASE-KEY-LONG im Schema-DDL-Eintrag für eine Satzart, die mit COPY in ein vom Anwenderprogramm verwendetes Subschema übernommen wird.
- Angabe der Klausel LOCATION MODE IS DIRECT-LONG im Schema-DDL-Eintrag für eine Satzart, die mit COPY in ein vom Anwenderprogramm verwendetes Subschema übernommen wird.
- Angabe der Klausel SET SELECTION THRU LOCATION im DDL-Set-Eintrag für eine Ownersatzart mit folgenden Eigenschaften:
  - Die Satzart ist mit LOCATION MODE IS DIRECT-LONG definiert.
  - Die Satzart wird mit COPY in ein vom Anwenderprogramm verwendetes Subschema übernommen.
- Definition eines Feldes mit USAGE IS DATABASE-KEY-LONG in einem vom Anwenderprogramm verwendeten Subschema.

Welche COBOL-DML- und CALL-DML-Anweisungen von diesen DDL-Klauseln betroffen sind, ist im nachfolgenden Abschnitt „DML-Anweisungen anpassen" beschrieben.

Für Subschemata, auf die sich Datentypänderungen (DATABASE-KEY-LONG) in der Schema-DDL auswirken, gelten nach der Neuübersetzung andere Validierungskriterien. Deshalb müssen Anwenderprogramme angepasst, neu übersetzt und gebunden werden, wenn sie solche Subschemata verwenden.

8.5.2 DML-Anweisungen anpassen

DML-Anweisungen, die einen Database-Key-Wert verwenden, liefern Hinweise, wo im Anwenderprogramm ggf. Anpassungen an Database-Key-Werte des erweiterten Wertebereichs (REC-REF > 254 und/oder RSQ > 2^{24}-1) notwendig sind.

Die folgenden COBOL-DML-Anweisungen verwenden einen Database-Key-Wert:

- ACCEPT
- FIND (Format 1) bzw. FETCH (Format 1)
- FIND (Format 7) bzw. FETCH (Format 7)
- MODIFY
- SET
- STORE

Die folgenden CALL-DML-Anweisungen verwenden einen Database-Key-Wert:

- ACCEPTC
- FIND1 bzw. FTCH1
- STORE1 bzw. STORE2

Die DML-Anweisungen sind ausführlich beschrieben im Handbuch „Anwendungen programmieren“.

8.5.2.1 Überblick

Tabelle 50 liefert ein grobes Entscheidungsraster, wann bei den einzelnen DML-Anweisungen Anpassungen an Database-Key-Werte des erweiterten Wertebereichs notwendig sind. In Tabelle 50 wird nur die Funktionalität der DML-Anweisungen betrachtet und nicht zwischen COBOL-DML und CALL-DML unterschieden.

DML-Anweisung	**... muss angepasst werden, wenn gilt:**		
ACCEPT FIND/FETCH-1	Zur von der Anweisung verwendeten Satzart gibt es Database-Key-Werte mit einer REC-REF > 254 und/oder RSQ > 2^{24}-1	–	–
STORE	–	DDL-Eintrag der verwendeten Satzart enthält die Klausel LOCATION MODE IS DIRECT-LONG	–
STORE, MODIFY, FIND/FETCH-7 (ohne CURRENT)	–	DDL-Eintrag der Ownersatzart enthält die Klausel LOCATION MODE IS DIRECT-LONG	DDL-Eintrag des betroffenen Set, in der die Satzart Owner ist, enthält die Klausel: SET SELECTION THRU LOC MODE OF OWNER

Tabelle 50: Kriterien für die Anpassung von DML-Anweisungen an erweiterte Database-Key-Werte

8.5.2.2 COBOL-DML-Anweisungen anpassen

ACCEPT

Die Anweisung ACCEPT gibt es in zwei Varianten:

- Im Format 1 ermittelt ACCEPT den Database-Key-Wert des CRR, CRS, CRA oder CRU und legt ihn in einem COBOL-Feld vom Typ USAGE IS DATABASE-KEY ab. Dieses Feld muss angepasst werden (USAGE IS DATABASE-KEY-LONG), falls nicht ausgeschlossen werden kann, dass die Database-Key-Werte von CRR, CRS, CRA bzw. CRU im erweiterten Wertebereich liegen.
  Wenn der ermittelte Database-Key-Wert zu groß für das Ergebnisfeld ist, meldet UDS/SQL den Statuscode 15102.
- Im Format 2 ermittelt ACCEPT den Realm, in dem der Satz liegt, dessen Database-Key-Wert im COBOL-Feld vom Typ USAGE IS DATABASE-KEY angegeben ist. Dieses Feld muss angepasst werden (USAGE IS DATABASE KEY LONG), wenn in ihm Database-Key-Werte aus dem erweiterten Wertebereich übergeben werden sollen.

FIND (Format 1) und FETCH (Format 1)

Die Anweisungen FIND und FETCH im Format 1 ermitteln einen Datenbanksatz über den Wert eines COBOL-Feldes vom Typ USAGE IS DATABASE-KEY. Dieses Feld muss angepasst werden (USAGE IS DATABASE-KEY-LONG), wenn nach einem Satz gesucht werden soll, dessen Database-Key-Wert im erweiterten Wertebereich liegt.

FIND (Format 7) und FETCH (Format 7)

Das Anwenderprogramm kann mit FIND/FETCH im Format 7 die gewünschte Set-Occurrence über den Database-Key-Wert des Ownersatzes auswählen, wenn die Schema-DDL die folgenden beiden Klauseln enthält:

- LOCATION MODE IS DIRECT bzw. LOCATION MODE IS DIRECT-LONG im Satz-Eintrag der Ownersatzart
- SET OCCURRENCE SELECTION THRU LOCATION MODE OF OWNER im betreffenden Set-Eintrag

Für die Auswahl der Set-Occurrence übergibt das Anwenderprogramm den Database-Key-Wert an das in der LOCATION MODE-Klausel genannte Database-Key-Feld (siehe Handbuch „Entwerfen und Definieren“).

Im Fall LOCATION MODE IS DIRECT-LONG hat das zugehörige Database-Key-Feld den Datentyp DATABASE-KEY-LONG. Damit das Anwenderprogramm Database-Key-Werte des erweiterten Wertebereichs übergeben kann, muss der Datentyp des COBOL-Feldes, das den Database-Key-Wert übergibt, geändert werden in USAGE IS DATABASE-KEY-LONG.

MODIFY

Das Anwenderprogramm kann mit MODIFY den Current Record of Rununit (CRU) in eine andere Set Occurrence umhängen, wenn die Schema-DDL die folgenden beiden Klauseln enthält:

- LOCATION MODE IS DIRECT bzw. LOCATION MODE IS DIRECT-LONG im Satz-Eintrag der Ownersatzart und
- SET OCCURRENCE SELECTION THRU LOCATION MODE OF OWNER im betreffenden Set-Eintrag,

Das Anwenderprogramm kann mit MODIFY die gewünschte Set-Occurrence über den Database-Key-Wert des Ownersatzes auswählen, indem es den Database-Key-Wert an das in der LOCATION MODE-Klausel genannte Database-Key-Feld übergibt (siehe Handbuch „Entwerfen und Definieren“).

Im Fall LOCATION MODE IS DIRECT-LONG hat das zugehörige Database-Key-Feld den Datentyp DATABASE-KEY-LONG. Damit das Anwenderprogramm Database-Key-Werte des erweiterten Wertebereichs übergeben kann, muss der Datentyp des COBOL-Feldes, das den Database-Key-Wert übergibt, geändert werden in USAGE IS DATABASE-KEY-LONG.

SET

Die Anweisung SET überträgt den Inhalt eines Database-Key-Feldes in ein anderes Database-Key-Feld bzw. in mehrere andere Database-Key-Felder.

Die Zielfelder müssen an den erweiterten Wertebereich angepasst werden (USAGE IS DATABASE-KEY-LONG), wenn Folgendes zutrifft:

- Das Feld, aus dem übertragen werden soll, wurde bereits in den Datentyp USAGE IS DATABASE-KEY-LONG geändert oder das Feld ist eine Redefinition eines Feldes vom Typ USAGE IS DATABASE-KEY-LONG.
- Es sollen Database-Key-Werte des erweiterten Wertebereichs übertragen werden.

Bei der Zuweisung von Database-Key-Werten mit der SET-Anweisung verfährt das COBOL-Laufzeitsystem wie folgt (siehe Handbuch „Anwendungen programmieren“):

Übertragung nach **von**	DATABASE-KEY	DATABASE-KEY-LONG
DATABASE-KEY	a	b
DATABASE-KEY-LONG	c	a

a. Werte werden 1:1 abgebildet
b. Werte werden intern verlängert
c. Werte werden intern verkürzt. Wenn der zu übertragende Database-Key-Wert zu groß für ein Zielfeld ist (REC-REF > 254 und/oder RSQ > 2^{24}-1), wird Statuscode 00102 gemeldet und der Wert des Zielfelds auf 0 gesetzt. Die Ausführung der Anweisung wird jedoch nicht abgebrochen.

STORE

Die Anweisung STORE nimmt einen neuen Satz in die Datenbank auf. Wenn der Schema-Eintrag für die betreffende Satzart die Klausel LOCATION MODE IS DIRECT bzw. LOCATION MODE IS DIRECT-LONG enthält, kann das Anwenderprogramm den Database-Key-Wert für den abzuspeichernden Satz vorgeben. Zu diesem Zweck übergibt das Anwenderprogramm den Database-Key-Wert an das in der LOCATION MODE-Klausel genannte Database-Key-Feld.

UDS/SQL verfährt dann wie nachfolgend beschrieben:

- Database-Key-Vergabe:
  UDS/SQL speichert den Satz unter dem vom Anwenderprogramm bereitgestellten Database-Key-Wert in der Datenbank ab.
- Set-Occurrence-Auswahl:
  Falls die betreffende Satzart Membersatzart eines Set ist, der mit SET OCCURRENCE SELECTION THRU LOCATION MODE OF OWNER definiert ist, speichert UDS/SQL den Satz als Membersatz ab in der Set-Occurrence desjenigen Ownersatzes, dessen Database-Key-Wert mit dem vom Anwenderprogramm bereitgestellten Database-Key-Wert übereinstimmt.

Im Fall LOCATION MODE IS DIRECT-LONG hat das zugehörige Database-Key-Feld den Datentyp DATABASE-KEY-LONG. Damit das Anwenderprogramm Database-Key-Werte des erweiterten Wertebereichs übergeben kann, muss der Datentyp des COBOL-Feldes, das den Wert übergibt, geändert werden in USAGE IS DATABASE-KEY-LONG.

8.5.2.3 CALL-DML-Anweisungen anpassen

ACCPTC

Die Anweisung ACCEPTC gibt es in zwei Varianten:

- Bei der Funktionswahl DB-KEY, DBKREC, DBKRLM oder DBKSET ermittelt ACCPTC den Database-Key-Wert des CRR, CRS, CRA bzw. CRU und stellt ihn im Benutzerinformationsbereich (UINF) zur Verfügung. Die Anweisung ACCPTC muss durch die Anweisung ACCPTL ersetzt werden, falls nicht ausgeschlossen werden kann, dass die Database-Key-Werte von CRR, CRS, CRA bzw. CRU im erweiterten Wertebereich liegen.
  Bei ACCPTC meldet UDS/SQL Statuscode 15102, wenn der ermittelte Database-Key-Wert im erweiterten Wertebereich liegt.
- Bei der Funktionsauswahl RLMDBK ermittelt ACCPTC den Realm, der den Satz mit dem im Benutzerinformationsbereich (UINF) angegebenen Database-Key-Wert enthält. Die Anweisung ACCPTC muss durch die Anweisung ACCPTL ersetzt werden, wenn im Benutzerinformationsbereich Database-Key-Werte aus dem erweiterten Wertebereich übergeben werden sollen.

FIND1 und FTCH1

Die Anweisungen FIND1 und FTCH1 ermitteln einen Datenbanksatz über einen Database-Key-Wert, den der Datenbankprogrammierer im Benutzerinformationsbereich (UINF) angibt. Die Anweisung FIND1 bzw. FTCH1 muss durch die Anweisung FIND1L bzw. FTCH1L ersetzt werden, wenn nach einem Satz gesucht werden soll, dessen Database-Key-Wert im erweiterten Wertebereich liegt.

STORE1 und STORE2

Die Anweisung STORE1 speichert einen vollständigen Satz in der Datenbank ab. Die Anweisung STORE2 speichert einzelne Felder in der Datenbank ab; STORE2 speichert die Felder komprimiert ab, wenn die betreffende Satzart in der SSL mit der Klausel COMPRESSION ALL definiert ist.
Wenn die betreffende Satzart in der Schema-DDL mit der Klausel LOCATION MODE IS DIRECT bzw. LOCATION MODE IS DIRECT-LONG vereinbart ist, speichert UDS/SQL den Satz unter dem Database-Key-Wert ab, den der Datenbankprogrammierer im Spezialparameter-2 (SPP2) angibt.
Im SPP2 der Anweisungen STORE1 und STORE2 können Database-Key-Werte des erweiterten Wertebereichs nicht angegeben werden. Damit im Fall LOCATION MODE IS DIRECT-LONG auch Database-Key-Werte des erweiterten Wertebereichs angegeben werden können, muss die Anweisung STORE1 bzw. STORE2 ersetzt werden durch STOR1L bzw. STOR2L.

8.5.3 COBOL-Definitionen anpassen

COBOL-Definitionen, die sich auf den Database Key beziehen, liefern Hinweise, wo im Anwenderprogramm ggf. Anpassungen an Database-Key-Werte des erweiterten Wertebereichs (REC-REF > 254 und/oder RSQ > 2^{24}-1) notwendig sind.

Die folgenden COBOL-Definitionen beziehen sich auf den Database Key:

- Definition eines COBOL-Feldes mit Datentyp USAGE IS DATABASE-KEY
- Redefinition mit REDEFINES eines COBOL-Feldes oder Subschema-Feldes vom Datentyp USAGE IS DATABASE KEY
- LINKAGE SECTION, die COBOL- oder Subschema-Felder vom Datentyp USAGE IS DATABASE-KEY beschreibt

Somit liefern diese Definitionen Hinweise, wo im Anwenderprogramm ggf. Anpassungen an erweiterte Database-Key-Werte notwendig sind.

Definitionen von COBOL-Feldern mit USAGE IS DATABASE-KEY anpassen

Der Definition eines Database-Key-Feldes im Subschema entspricht die Definition von COBOL-Feldern des Typs USAGE IS DATABASE-KEY bzw. USAGE IS DATABASE-KEY-LONG in der WORKING-STORAGE SECTION des COBOL-Programms.

Ein COBOL-Feld, das mit USAGE IS DATABASE-KEY definiert ist, zeigt an, dass das Anwenderprogramm Database-Key-Werte verwendet. Sofern dieses COBOL-Feld auch Database-Key-Werte des erweiterten Wertebereichs aufnehmen soll (z.B. mit ACCEPT), muss der Datenyp des COBOL-Feldes geändert werden in USAGE IS DATABASE-KEY-LONG.

Wenn ein COBOL-Programm nur Database-Key-Felder (USAGE IS DATABASE-KEY) verwendet, die innerhalb der WORKING STORAGE SECTION des Programms definiert sind, kann das COBOL-Programm bereits vor der Datenbankumstellung für die Nutzung erweiterter Database-Key-Werte (USAGE IS DATABASE-KEY-LONG) vorbereitet werden.

Redefinitionen von Database-Key-Feldern anpassen

Die Redefinition von Database-Key-Feldern mit REDEFINES in der WORKING-STORAGE SECTION erlaubt es, einzelne Bestandteile von Database-Key-Werten im COBOL-Programm gezielt anzusprechen und zu verarbeiten. Redefiniert werden können folgende Database-Key-Felder:

- COBOL-Felder vom Datentyp USAGE IS DATABASE-KEY bzw.
  USAGE IS DATABASE-KEY-LONG
- Felder vom Datentyp USAGE IS DATABASE-KEY bzw. USAGE IS DATABASE-KEY-LONG, die in einem vom Anwenderprogramm verwendeten Subschema (SUB-SCHEMA SECTION) definiert sind.

Für COBOL- oder Subschema-Felder vom Datentyp USAGE IS DATABASE-KEY-LONG müssen alle Redefinitionen entsprechend angepasst werden.

LINKAGE SECTION anpassen

Die LINKAGE SECTION in einem aufgerufenen COBOL-Programm beschreibt und verwendet eine Datenstruktur, die außerhalb des Programms in einem aufrufenden Programm definiert ist.

Wenn in der LINKAGE SECTION eines Anwenderprogramms ein Subschema- oder COBOL-Feld vom Typ USAGE IS DATABASE-KEY angesprochen wird, ist zu prüfen,

- ob und ggf. wie dieses Feld mit Werten versorgt wird,
- ob dieses Feld mit Werten versorgt wird, die aus Sätzen anderer Datenbanken stammen,
- ob bei der Wertzuweisung der Datentyp konvertiert wird.

Wenn das im aufrufenden Programm definierte Feld in den Datentyp USAGE IS DATABASE-KEY-LONG geändert wurde und nicht ausgeschlossen werden kann, dass künftig auch Database-Key-Werte des erweiterten Wertebereichs übergeben werden, muss die lokale Definition ebenfalls in DATABASE-KEY-LONG geändert werden.

In der Regel verweist die LINKAGE-SECTION per COPY-Anweisung auf ein im aufrufenden Programm definiertes Datenfeld. Um Datentypänderungen des aufrufenden Programms in das aufgerufene Programm zu übernehmen, genügt es dann, das aufgerufene Programm neu zu übersetzen.

8.5.4 Weitere Stellen im Anwenderprogramm anpassen

Das Anwenderprogramm muss auch dort an die Nutzung erweiterter Database-Key-Werte angepasst werden, wo sich die Verwendung von Database-Key-Werten nicht ohne weiteres erkennen lässt. Dies betrifft folgende Fälle:

- Das Anwenderprogramm ermittelt Database-Key-Werte aus Anwenderdaten.
- Das Anwenderprogramm überträgt Database-Key-Werte zwischen zwei oder mehreren Satzarten.
- Das Anwenderprogramm verwendet Database-Key-Werte datenbankübergreifend. In diesem Fall können neben Änderungen im Anwenderprogramm auch Anpassungen in den beteiligten Datenbanken erforderlich sein (siehe "Kriterien für die Umstellung").
- Das Anwenderprogramm verwaltet Set-Verbindungen über Database-Key-Werte.
- Das Anwenderprogramm verwendet Database-Key-Werte,
  - um die Verwaltung der Sätze dieser Satzart direkt zu beeinflussen,
  - um sich Verknüpfungen und Abhängigkeiten zu merken.
- Das Anwenderprogramm verwendet ein COBOL-Feld, das nicht vom Datentyp USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG ist, um Database-Key-Werte zwischenzuspeichern oder zu verarbeiten.
- Das Anwenderprogramm verwendet ein COBOL-Feld, um Database-Key-Werte unterschiedlicher Satzarten zwischenzuspeichern oder zu verarbeiten.
- Das Anwenderprogramm überträgt Database-Key-Werte in Datenbankfelder, die nicht vom Typ DATABASE-KEY oder DATABASE-KEY-LONG sind. In diesem Fall können neben Änderungen im Anwenderprogramm auch Anpassungen in der betroffenen Datenbank erforderlich sein (siehe "Kriterien für die Umstellung").

8.6 SQL-, IQS- und KDBS-Anwendungen anpassen

SQL-Anwendungen anpassen

Anwenderprogramme können mit SQL auf UDS/SQL-Datenbanken zugreifen.

Wenn ein SQL-Programm Database-Key-Werte verarbeitet, ist zu prüfen, ob die betroffenen Database Keys vom Datentyp DATABASE-KEY-LONG sind. In diesem Fall müssen die Felder der Host-Sprache, die Database-Key-Werte aufnehmen, vom Datentyp CHARACTER(8) sein.

Primär- und Fremdschlüsselattribute bleiben von Umstellungen unbeeinflusst; sie behalten den Datentyp INTEGER.

IQS- und KDBS-Anwendungen anpassen

Entsprechend den CALL-DML- und COBOL-DML-Anwendungen sind auch Anwendungen des Dialogsystems IQS sowie der kompatiblen Datenbankschnittstelle UDS-KDBS zu überprüfen, ob in ihnen direkt auf Database-Key-Felder zugegriffen wird.

Wenn eine Anwendung auf ein Database-Key-Feld vom Typ DATABASE-KEY-LONG zugreift, müssen die betroffenen Datenfelder in der Anwendung hinsichtlich des Datentyps und der Versorgung mit Werten angepasst werden.

8.7 Beispiele zur Datenbankumstellung

Die folgenden Beispiele skizzieren den Umstellvorgang für zwei unterschiedliche Anwenderszenarien:

- Database-Key-Erweiterung bei transaktionsübergreifender Verwendung von Database-Key-Werten innerhalb *einer* Datenbank
- Database-Key-Erweiterung bei datenbankübergreifender Verwendung von Database-Key-Werten in einer Multi-DB-Konfiguration

Die Beispiele beziehen sich auf die Datenbanken REISEN, KUNDEN und VERSAND. In den Beispielen wird unterstellt, dass die Datenbanken REISEN, KUNDEN und VERSAND zunächst im 2-Kbyte-Format vorliegen. Nach dem in den Beispielen skizzierten Umstellvorgang sind die Datenbanken REISEN, KUNDEN und VERSAND identisch mit den im Abschnitt „Beispieldatenbanken" beschriebenen 4-Kbyte-Datenbanken. Die in den Beispielen angeführten Anwenderprogramme AP1, AP2 und AP3 sind COBOL-Programme.

8.7.1 Transaktionsübergreifende Nutzung von erweiterten Database-Key-Werten

Die transaktionsübergreifende Nutzung von Database-Key-Werten innerhalb *einer* Datenbank ist die typische Situation für die Verwendung von Database-Key-Werten in Anwenderprogrammen:

1. In einer Transaktion TA1 ermittelt das Anwenderprogramm AP1 mit der DML-Anweisung ACCEPT den Database-Key-Wert des CRR der Satzart KUNDE und speichert diesen Wert in einem COBOL-Feld DBK vom Datentyp USAGE IS DATABASE-KEY:

   ```
   ACCEPT DBK FROM KUNDE CURRENCY
   ```

   Die Satzart KUNDE ist definiert im Schema REISEBUERO der Datenbank REISEN.
2. In einer nachfolgenden Transaktion TA2 positioniert das Anwenderprogramm AP1 mit der DML-Anweisung FIND (Format 1) sofort wieder auf den KUNDE-Satz mit dem im COBOL-Feld DBK gespeicherten Database-Key-Wert:

   ```
   FIND KUNDE DATABASE-KEY IS DBK
   ```

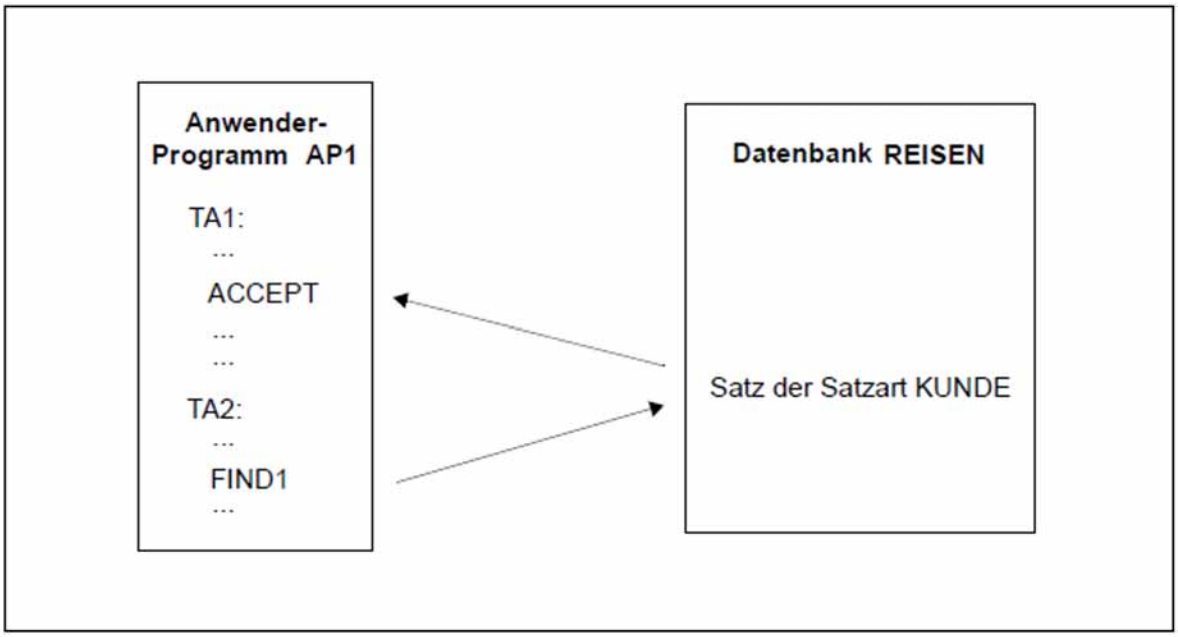

Bild 39: Transaktionsübergreifende Nutzung von Database-Key-Werten durch ein Anwenderprogramm

In der 2-Kbyte-Datenbank REISEN können zur Satzart KUNDE maximal 16 777 215 ($= 2^{24}$-1) Sätze gespeichert werden. Künftig soll es zur Satzart KUNDE mehr als 16 777 215 Sätze geben. Entsprechend soll das Anwenderprogramm AP1 die erweiterten Database-Key-Werte verarbeiten können. Andere Kapazitätsengpässe treten in der Datenbank REISEN nicht auf.

Die notwendigen Umstellungs- und Anpassungsmaßnahmen führen Sie in folgenden Schritten durch:

1. Mit dem Dienstprogramm BPGSIZE stellen Sie die Datenbank REISEN auf das 4-Kbyte-Format um. Anschließend erklären Sie die umgestellte Datenbank zum Original.
2. Sie bereiten die Datenbank REISEN für den Datenbankbetrieb vor. Insbesondere erweitern Sie mit dem Dienstprogramm BREORG im Schema REISEBUERO das Mengengerüst für die Satzart KUNDE auf den benötigten Wert.
   Zur Satzart KUNDE können nun mehr als 16 777 215 Sätze in der Datenbank REISEN abgespeichert werden.
3. Da Database-Key-Werte für die Satzart KUNDE nicht vom Anwenderprogramm vergeben werden (keine LOCATION MODE IS DIRECT-Klausel im DDL-Satzeintrag für die Satzart KUNDE), müssen Sie Schema-DDL

(und somit Subschema-DDL) für das Schema REISEBUERO der Datenbank REISEN nicht anpassen; die Datenbank REISEN muss also nicht umstrukturiert werden.

4. Bereits vorhandene Anwenderprogramme sind weiterhin unverändert ablauffähig, solange sie nicht in der Datenbank REISEN auf KUNDE-Sätze zugreifen, deren Database-Key-Werte eine RSQ > 2^{24}-1 enthalten.
   Damit das Anwenderprogramm AP1 in der Datenbank REISEN auch auf KUNDE-Sätze zugreifen kann, deren Database-Key-Werte eine RSQ > 2^{24}-1 enthalten, muss die Deklaration des COBOL-Feldes DBK im Anwenderprogramms AP1 angepasst werden in den Datentyp USAGE IS DATABASE-KEY-LONG. Dabei sind ggf. auch Redefinitionen des Feldes DBK zu berücksichtigen.
   Das Anwenderprogramm AP1 muss neu übersetzt und gebunden werden.
   Die Anpassung des Anwenderprogramms ist zeitlich nicht an die Umstellung der Datenbank gebunden.

8.7.2 Database-Key-Erweiterung in einer Multi-DB-Konfiguration

Dieses Beispiel skizziert eine Multi-DB-Konfiguration, in der ein Anwenderprogramm AP2 über *einen* UDS/SQL-DBH auf die Datenbanken KUNDEN und VERSAND zugreift.
Das Anwenderprogramm AP2 enthält zwei Module MODUL1 und MODUL2, die Database-Key-Werte austauschen über ein global definiertes COBOL-Feld DBK vom Datentyp USAGE IS DATABASE-KEY. MODUL1 greift auf die Datenbank KUNDEN zu, MODUL2 greift auf die Datenbank VERSAND zu (siehe Bild 40):

1. MODUL1 ermittelt mit der DML-Anweisung ACCEPT in der Datenbank KUNDEN den Database-Key-Wert des CRR der Satzart KUNDE und speichert diesen Wert in einem global definierten COBOL-Feld DBK vom Typ USAGE IS DATABASE-KEY:

   ```
   ACCEPT DBK FROM KUNDE CURRENCY
   ```

   Die Satzart KUNDE ist definiert im Schema KUNDENDATEI der Datenbank KUNDEN. Außerdem ist die Satzart KUNDE in einem Subschema von KUNDENDATEI enthalten, das von MODUL2 verwendet wird (Übernahme mit COPY KUNDE in der Subschema-DDL).

2. MODUL2 versorgt im Satzbereich der UWA (User Work Area) für die Datenbank VERSAND das Feld KUNDEN-NR der Satzart KUNDE mit dem im COBOL-Feld DBK gespeicherten Database-Key-Wert:

   ```
   SET KUNDEN-NR TO DBK
   ```

   Die Satzart KUNDE ist definiert im Schema ARTIKELVERSAND der 2-Kbyte-Datenbank VERSAND, das Feld KUNDEN-NR hat den Datentyp DATABASE-KEY. Außerdem ist die Satzart KUNDE in einem Subschema von ARTIKELVERSAND enthalten, das von MODUL2 verwendet wird (Übernahme mit COPY KUNDE in der Subschema-DDL).

3. Nachdem auch die restlichen Felder des KUNDE-Satzes in der UWA von MODUL2 mit Werten versorgt sind, überträgt MODUL2 diesen Satz in die Datenbank VERSAND.

   ```
   STORE KUNDE [...]
   ```

   Da in der Schema-DDL für die 2-Kbyte-Datenbank VERSAND der Eintrag für die Satzart KUNDE die Klausel LOCATION MODE IS DIRECT KUNDEN-NR OF KUNDE enthält, ist der im Feld KUNDEN-NR abgelegte Wert gleichzeitig der Database-Key-Wert des abgespeicherten Satzes.

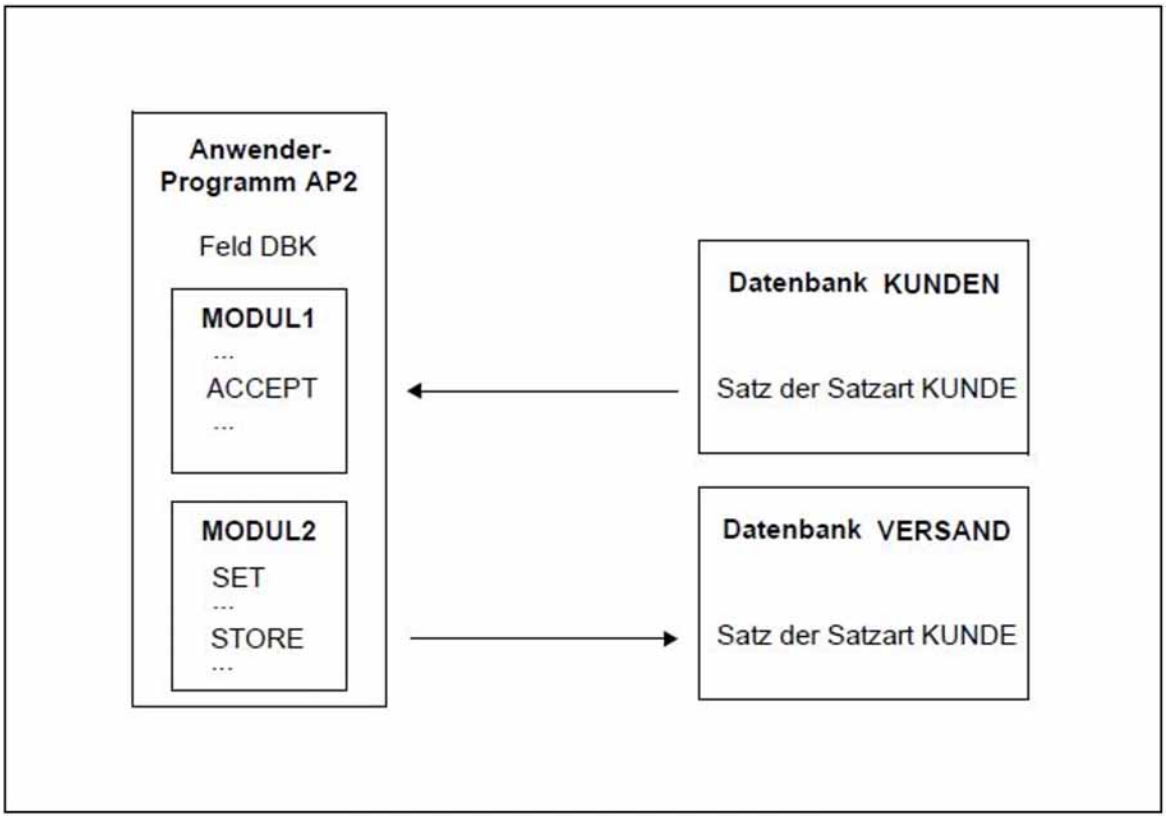

Bild 40: Datenbankübergreifende Nutzung von Database-Key-Werten durch ein Anwenderprogramm

Sowohl zur Satzart KUNDE im Schema KUNDENDATEI der 2-Kbyte-Datenbank KUNDEN als auch zur Satzart KUNDE im Schema ARTIKELVERSAND der 2-Kbyte-Datenbank VERSAND können zunächst nur maximal 16 777 215 (=2^{24}-1) Sätze gespeichert werden.

Zur Satzart KUNDE im Schema KUNDENDATEI der Datenbank KUNDEN soll es künftig mehr als 16 777 215 Sätze geben. Andere Kapazitätsengpässe treten in der Datenbank KUNDEN nicht auf.

Die Datenbanken KUNDEN und VERSAND können Sie unabhängig voneinander umstellen und anpassen. Es empfiehlt sich folgende Reihenfolge:

Datenbank VERSAND umstellen und umstrukturieren

1. Mit dem Dienstprogramm BPGSIZE stellen Sie die 2-Kbyte-Datenbank VERSAND auf das 4-Kbyte_Format um. Die Umstellung ist notwendig, da künftig Database-Key-Wer-te mit einer RSQ > 2^{24}-1 aus der Datenbank KUNDEN in die Satzart KUNDE der Datenbank VERSAND übertragen werden.
   Anschließend erklären Sie die umgestellte Datenbank zum Original.
2. Sie bereiten die Datenbank VERSAND für den Datenbankbetrieb vor. Insbesondere erweitern Sie mit dem Dienstprogramm BREORG im Schema ARTIKELVERSAND das Mengengerüst für die Satzart KUNDE auf den benötigten Wert.
   Zur Satzart KUNDE des Schemas ARTIKELVERSAND können nun mehr als 2^{24}-1 Sätze in der Datenbank VERSAND abgespeichert werden.
3. Sie strukturieren die Datenbank VERSAND um:
   - Im Satzeintrag der Satzart KUNDE im Schema ARTIKELVERSAND ändern Sie die Klausel LOCATION MODE IS DIRECT KUNDEN-NR OF KUNDE in LOCATION MODE IS DIRECT-LONG KUNDEN-NR OF KUNDE. Den Datentyp des Feldes KUNDEN-NR ändern Sie in den Datentyp DATABASE-KEY-LONG.

- Das von MODUL2 benötigte Subschema müssen Sie nicht ändern, da es die Satzart KUNDE aus dem SCHEMA ARTIKELVERSAND mit COPY übernimmt. Das Subschema muss nur neu übersetzt werden. Außerdem müssen alle Anwenderprogramme, die dieses Subschema verwenden, neu übersetzt und gebunden werden.
- Die weiteren Umstrukturierungsmaßnahmen führen Sie durch gemäß dem Kapitel „Datenbank umstrukturieren (BCHANGE, BALTER)".

Datenbank KUNDEN umstellen

1. Mit dem Dienstprogramm BPGSIZE stellen Sie die 2-Kbyte-Datenbank KUNDEN auf das 4-Kbyte-Format um. Anschließend erklären Sie die umgestellte Datenbank zum Original.
2. Sie bereiten die Datenbank KUNDEN für den Datenbankbetrieb vor. Insbesondere erweitern Sie mit dem Dienstprogramm BREORG im Schema KUNDENDATEI das Mengengerüst für die Satzart KUNDE auf den benötigten Wert.
   Zur Satzart KUNDE des Schemas KUNDENDATEI können nun mehr als 2^{24}-1 Sätze in der Datenbank KUNDEN abgespeichert werden.
3. Im Schema KUNDENDATEI der Datenbank KUNDEN sind keine Änderungen erforderlich, die Datenbank KUNDEN muss also nicht umstrukturiert werden.

Anwenderprogramm AWP2 anpassen

Im Anwenderprogramm AWP2 muss die Deklaration des COBOL-Feldes DBK geändert werden in den Datentyp USAGE IS DATABASE-KEY-LONG. Dabei sind ggf. auch Redefinitionen des Feldes DBK zu berücksichtigen. Das Anwenderprogramm AWP2 muss neu übersetzt und gebunden werden.

Alternative Vorgehensweise: zeitlich entkoppelt umstellen und umstrukturieren

Alternativ zum oben geschilderten Vorgehen können Sie die Umstellung der Datenbanken mit BPGSIZE und die Umstrukturierung jeweils zeitlich entkoppelt durchführen, z.B.:

1. Datenbank VERSAND mit BPGSIZE umstellen; zunächst nicht umstrukturieren
2. Datenbank KUNDEN mit BPGSIZE umstellen; zunächst nicht umstrukturieren
3. Datenbank VERSAND umstrukturieren und Anwenderprogramme anpassen
4. Datenbank KUNDEN umstrukturieren und Anwenderprogramme anpassen

Zwischen den einzelnen Schritten ist normaler Datenbankbetrieb möglich. Da zumindest einige Anwenderprogramme sowohl bei 3.) als auch bei 4.) angepasst und übersetzt werden müssen, bietet es sich an, die Schritte 3.) und 4.) zusammenzufassen.

9 Umsetzung von Datenbanken auf DB-Layout-Version 4 (BTRANS24)

Für den Versionsumstieg aus UDS/SQL der Versionen V2.0 bis V2.3 müssen die Datenbanken mit dem Dienstprogramm BTRANS24 auf die Datenbank-Layout-Version '004.00' umgesetzt werden.

Auch Datenbanken, die letztmalig mit UDS/SQL-Versionen bis einschließlich V2.3 bearbeitet wurden, müssen vor einer Verarbeitung - etwa für Revisionszwecke - mit UDS/SQL-Versionen ab V2.4 auf die Datenbank-Layout-Version '004.00' umgesetzt werden. Mit der Umsetzung wird auch die Realm-Layout-Version aller Realms auf '004.00' umgesetzt.

Das Dienstprogramm BTRANS24 ist Bestandteil der Liefereinheit UDS-SQL-T und steht standardmäßig in der Bibliothek SIPPRG.UDS-SQL-T.029 zur Verfügung.

Das Dienstprogramm BTRANS24 stellt für die Umsetzung der DB-Layout-Version zwei Funktionen zur Verfügung:

- "Voraussetzungen für die Umsetzung prüfen"
- Datenbankumsetzung durchführen (siehe "Datenbankumsetzung mit BTRANS24 durchführen")

Eine Beschreibung der Anweisungen von BTRANS24 finden Sie in Abschnitt „Anweisungen für BTRANS24".

Wie Sie BTRANS24 aufrufen, ist im Abschnitt „BTRANS24 aufrufen" beschrieben.

Die Meldungen, die BTRANS24 ausgibt, finden Sie im Handbuch „Meldungen".

9.1 Voraussetzungen für die Umsetzung prüfen

Mit dem Dienstprogramm BTRANS24 können Sie prüfen, ob die Voraussetzungen zur Umsetzung einer Datenbank in das Layout der Version '004.00' erfüllt sind.

Einen Prüflauf können Sie mit der Anweisung CHECK-DATABASE explizit anfordern. Dann ist er auch parallel zur laufenden DBH-Session einsetzbar. Außerdem wird vor der eigentlichen Umsetzung der Datenbank (Anweisung TRANSFORM-DATABASE) immer implizit ein Prüflauf durchgeführt.

Folgende Voraussetzungen müssen für die Umsetzung erfüllt sein:

- Die umzusetzenden Datenbanken müssen Originaldatenbanken sein.
- Die umzusetzenden Datenbanken müssen in der Datenbank-Layout-Version '002.00' oder '003.00' vorliegen; sie müssen also mindestens auf die Nutzung mit UDS/SQL V2.0 umgestellt sein.
  Ist dies nicht der Fall, dann müssen die Datenbanken zuerst mit den Umsetzdienstprogrammen der entsprechenden UDS/SQL-Versionen umgesetzt werden (BTRANS20 für die Nutzung in UDS/SQL V2.0; BTRANSDB für die Nutzung in UDS/SQL V1.2).
- Für jede DBTT muss in dem entsprechenden Realm eine leere Datenbankseite zur Verfügung stehen, in der der Anfangsstand der DBTT-Ankertabelle abgelegt werden kann. Ist dies nicht der Fall, so muss durch Offline-Realm-Erweiterung mit dem Dienstprogramm BREORG **der Version V2.3** genügend freier Platz im Realm bereitgestellt werden.
  Das Dienstprogramm BREORG der Version 2.3 steht auch in der aktuellen UDS/SQL-Version zur Verfügung. Es ist Bestandteil der Liefereinheit UDS-SQL-T und standardmäßig in der Bibliothek SIPPRG.UDS-SQL-T.029 enthalten. Für BREORG V2.3 gibt es kein Start-Kommando; das Dienstprogramm kann mit folgendem Kommando gestartet werden:

  ```
  /START-EXECUTABLE-PROGRAM FROM-FILE=(LIB=UDS/SQL-T-modulbibliothek, ELEM=BREORG)
  ```

  Auch für DBTTs des PRIVACY-AND-IQF-Schemas und des Compiler-Schemas müssen freie Seiten im DBDIR (8 Seiten) und DBCOM (29 Seiten) zur Verfügung stehen.
- Sämtliche Realms der Datenbank müssen zugreifbar und zugeschaltet sein - insbesondere auch DBCOM.
- COSSD und HASHLIB werden für die Umsetzung nicht benötigt.

9.2 Datenbankumsetzung mit BTRANS24 durchführen

Die eigentliche Umsetzung einer Datenbank mit BTRANS24 erfolgt offline auf einem konsistenten Datenbestand. Die Umsetzung muss in der Datenbankkennung durchgeführt werden. Sie starten die Umsetzung der Datenbank mit der Anweisung TRANSFORM-DATABASE.

Prüflauf

Vor der Umsetzung wird implizit ein Prüflauf durchgeführt. Erst wenn in diesem Prüflauf festgestellt wird, dass die Voraussetzungen zur Datenbankumsetzung vollständig erfüllt sind, werden die Änderungen durchgeführt. Sind die Voraussetzungen nicht vollständig erfüllt, dann wird die Datenbank nicht umgesetzt. Die Datenbank ist weiterhin konsistent. Die Datenbank ist auch konsistent, wenn BTRANS24 bis zu diesem Zeitpunkt - aus welchen Gründen auch immer - abbricht.

Logging

Um den Versionsumstieg aus UDS/SQL V2.3 und früheren Versionen zu erleichtern, erfolgen alle Änderungen von BTRANS24 mit Alogging, wenn dieses eingeschaltet ist. Wie jedes ändernde Dienstprogramm, das Alogging unterstützt, erzeugt auch BTRANS24 zunächst einen Checkpoint für das AFIM-Logging. Wenn das Nachfahren z.B. wegen Platten-Crash in einer anschließenden Session mit einer höheren UDS/SQL-Version als V2.3 notwendig wird, kann die Datenbank auf dem Stand der letzten Sicherungskopie mit folgenden Hilfsmitteln rekonstruiert werden:

- dem BMEND von UDS/SQL V2.3 bzw. der vorher genutzten Version mittels UPDATE-DATABASE...DEADLINE=BREAKPOINT
- den ALOG-Dateien, die vor dem Umstieg entstanden sind
- der ALOG-Datei von BTRANS24
- den im weiteren Betrieb mit der höheren UDS/SQL-Version als V2.3 entstandenen ALOG-Dateien

Umsetzung

BTRANS24 führt folgende Änderungen durch:

- Eine neue, zu früheren Versionen inkompatible DB-Layout-Version wird im DBDIR eingetragen.
- Die benötigten DBTT-Ankertabellenseiten in den Realms werden erzeugt.
- Pro Satzart wird der ACTKEY der ersten neuen DBTT-Ankertabellenseite in die SIA eingetragen.
- Die neue DB_LAYOUT_VERSION wird im ACT_KEY0 und ACT_KEYN jedes Realms eingetragen.
- Falls ein Realm mit Realm-Layout-Version '002.00' vorliegt, werden im ACT_KEY0 die Verwaltungstabellen für FPA-Extents angelegt. Ungenutzte Datenbereiche des ACT_KEY0 werden gelöscht.
- Im ACT_KEY0 werden Datenbereiche zur wiederanlauffähigen Überwachung der Online-DBTT-Erweiterung initialisiert.

Insgesamt sind nur sehr wenige Änderungen in der Datenbank erforderlich. Daher ist die Laufzeit des Dienstprogrammes BTRANS24 kurz.

Das Dienstprogramm ist in der Änderungsphase nicht wiederanlauffähig. Beim Absturz muss auf einen evtl. mit ALOG-Dateien nachgefahrenen Sicherungsstand wiederaufgesetzt werden.

Es erfolgt keine Änderung des COSSD und der SSIAs im DBDIR. Das Kriterium zur Subschema-Validierung ändert sich nicht. Neuübersetzung oder Neubinden von Anwendungen ist daher nicht notwendig.

9.3 Anweisungen für BTRANS24

Für das Dienstprogramm BTRANS24 gibt es folgende SDF-Anweisungen:

Anweisung	Funktion
CHECK-DATABASE	Prüflauf anstoßen
TRANSFORM-DATABASE	Datenbank umsetzen
END	Eingabe der Anweisungen beenden und Ausführung starten

Tabelle 51: Anweisungen für BTRANS24

Sie müssen genau eine der beiden Anweisungen CHECK-DATABASE bzw. TRANSFORM-DATABASE angeben. Wenn Sie mehr als eine dieser Anweisungen jeweils richtig angeben, so ist die letzte dieser Anweisungen die gültige.

9.3.1 CHECK-DATABASE (Prüflauf anstoßen)

Mit der Anweisung CHECK-DATABASE stoßen Sie einen expliziten Prüflauf an. Der Prüflauf muss in der Datenbankkennung durchgeführt werden. Er kann parallel zur DBH-Session durchgeführt werden. Die Realms werden nur lesend eröffnet.

CHECK-DATABASE
DATABASE-NAME=<dbname>

DATABASE-NAME=<dbname>
Name der Datenbank.

CHECK-DATABASE bewertet die Voraussetzungen zur Umsetzung der Datenbank und gibt entsprechende Meldungen aus. Bitte beachten Sie, dass diese Bewertung fehlerhaft sein kann durch Änderungen, die erst nach dem Prüflauf in der Datenbank wirksam werden.

9.3.2 TRANSFORM-DATABASE (Datenbank umsetzen)

Mit dieser Anweisung stoßen Sie die Umsetzung der Datenbank an. Die Umsetzung muss in der Datenbankkennung durchgeführt werden.

TRANSFORM-DATABASE
DATABASE-NAME=<dbname>

DATABASE-NAME=<dbname>
Name der Datenbank.

Die Realms werden exklusiv zur Änderung geöffnet. Implizit wird zunächst ein Prüflauf durchgeführt. Falls hierbei festgestellt wird, dass die Voraussetzungen für eine Umsetzung nicht gegeben sind, wird die Umsetzung abgebrochen. Bis zu diesem Zeitpunkt wurden noch keine Änderungen in der Datenbank vorgenommen, die Datenbank ist weiterhin konsistent.

9.3.3 END (Eingabe der Anweisungen beenden)

Mit der Anweisung END beenden Sie die Eingabe der Anweisungen. Die Ausführung wird gestartet.

END

Diese Anweisung hat keine Operanden.

9.4 BTRANS24 aufrufen

BTRANS24 rufen Sie mit dem folgenden Kommando auf:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=(LIB=UDS/SQL-T-modulbibliothek,
                                     ELEM=BTRANS24)
```

BTRANS24 ist Bestandteil der Liefereinheit UDS-SQL-T und steht standardmäßig in der Bibliothek SIPPRG.UDS-SQL-T.029 zur Verfügung.

Anschließend können Sie die BTRANS24-Anweisungen eingeben, z.B.

```
//TRANSFORM-DATABASE DATABASE-NAME=dbname
//END
```

10 Fachwörter

Dieses Fachwortverzeichnis enthält Definitionen wichtiger Begriffe, die in den Handbüchern zu UDS/SQL verwendet werden.
Kursiv gedruckte Fachwörter in den Definitionen verweisen auf entsprechende Definitionen für diese Fachwörter.
Ein „siehe“-Verweis für ein Fachwort verweist auf das in den UDS/SQL-Handbüchern hauptsächlich verwendete Fachwort.

A

Act-Key
act-key

(actual key) Aktuelle Adresse einer *Seite*, bestehend aus *Realmnummer* und *Seitennummer*.

Act-Key-0-Seite
act-key-0 page

Erste *Seite* eines *Realm*. Sie enthält allgemeine Informationen über den Realm, z.B.

- Erstellungszeitpunkt des Realm,
- Zeitpunkt der letzten Änderung,
- *interne Versionsnummer* des Realm,
- Unterbrechungsinformationen des Systems (*Systembreak-Informationen*),
- ggf. Kenndaten für den *Warmstart*.

Act-Key-N-Seite
act-key-N page

Kennseite eines *Realm* mit der höchsten *Seitennummer*.
Kopie der *Act-Key-0-Seite*.

Administratortask
administrator task

Task des *independent DBH*. Der *Datenbankadministrator* kann über diese Task den Ablauf des *independent DBH* steuern.

Adresse, physische
address, physical

siehe *Act-Key* oder *Probable Position Pointer (PPP)*

Adressliste
pointer array

Tabelle, die auf die *Membersätze* einer *Set-Occurrence* verweist. Dient dem *sequentiellen* und *direkten Zugriff* auf die Membersätze.

AFIM
AFIM

siehe *After-Image*

After-Image
after-image

Geänderter Teil einer *Seite* **nach** einer Änderung des Seiteninhalts. After-Images schreibt der *DBH* sowohl in die *RLOG-Datei* als auch in die *ALOG-Datei.*

After-Image, ALOG-Datei
after-image, ALOG file

Die After-Images werden in die ALOG-Datei geschrieben, wenn der ALOG-Puffer voll ist. Die After-Images in der ALOG-Datei werden zur Langzeitsicherung, d.h. für lange Zeit benötigt. Sie werden benutzt, um eine Originaldatenbank zu rekonstruieren oder eine *Schattendatenbank* zu aktualisieren.

After-Image, RLOG-Datei
after-image, RLOG file

Die After-Images werden in die RLOG-Datei geschrieben, **bevor** die Änderungen auf der *Datenbank* festgeschrieben werden. Die After-Images in der RLOG-Datei werden nur zum *Warmstart* benötigt und deshalb zyklisch überschrieben.

ALOG-Datei
ALOG file

Datei zur Langzeitsicherung, siehe *After-Image*.

ALOG-Folgenummer
ALOG sequence number

Kennzeichnung im Dateinamen der *ALOG-Dateien* (000000001 - 999999999). Die erste ALOG-Datei einer *Datenbank* trägt immer die Folgenummer 000000001.

Ankersatz
anchor record

Satz, den UDS/SQL automatisch als *Ownersatz* für *SYSTEM-Sets* einrichtet. Er enthält keine mit der *Schema-DDL* definierten *Felder* und es kann auf ihn nicht zugegriffen werden.

Anweisungscode
statement code

Nummer, die im ersten Teil des Feldes *DATABASE-STATUS* hinterlegt wird und die darüber informiert, bei welcher *DML*-Anweisung ein Sonderzustand aufgetreten ist.

Anwenderprogramm (AP)
application program (AP)

Z.B. *COBOL-DML*-Programm, IQS.

Anwendertask
user task

Ausführung eines *Anwenderprogramms* bzw. *openUTM*-Teilprogramms, einschließlich der vom System dazugebundenen Teile.

Anwendung
application

Umsetzung einer Aufgabenstellung in ein *Anwenderprogramm* oder mehrere Anwenderprogramme, die mit UDS/SQL-*Datenbanken* arbeiten.

Area
area

siehe *Realm*

Ascending-Key (ASC-Key)
ascending key (ASC key)

Primärschlüssel eines *Set*. Der Ascending-Key legt die Reihenfolge der *Membersätze* in den *Set-Occurrences* nach aufsteigenden Schlüsselwerten fest.

Auftrag
request

Die Funktionen, die durch die *DAL*-Kommandos ADD DB, ADD RN, DROP DB, DROP RN, NEW RLOG und CHECKPOINT zunächst im *DBH* nur vorgemerkt sind, werden erst durch das DAL-Kommando PERFORM zur Durchführung angestoßen.

automatische DBTT-Erweiterung
automatic DBTT extension

Einige Dienstprogramme erweitern die Anzahl möglicher Sätze einer Satzart bei Engpässen automatisch; hierfür ist keine gesonderte Administration erforderlich.
Siehe auch *Online-DBTT-Erweiterung*.

automatische Realm-Erweiterung
automatic realm extension

Einige Dienstprogramme erweitern Realms bei Engpässen automatisch; hierfür ist keine gesonderte Administration erforderlich.
Siehe auch *Online-Realm-Erweiterung*.

B

Base Interface Block
Base Interface Block

siehe *BIB*

Before-Image
before-image

Teil einer *Seite* vor einer Änderung des Seiteninhalts.
Before-Images schreibt der *DBH* in die *RLOG-Dateien*. Dort werden die Before-Images während des Datenbankbetriebs geschrieben, bevor die Änderungen auf der *Datenbank* festgeschrieben werden. Voraussetzung ist, dass RLOG-Dateien geführt werden.

Benutzerdatenbank
user database

Die *Realms* und Dateien der *Datenbank*, die der Anwender benötigt, um Daten in die Datenbank zu speichern und wiederzugewinnen.
Dies sind:

- das *Database Directory (DBDIR)*
- die *Benutzerrealms*
- die Modulbibliothek für *Hashroutinen (HASHLIB)*.

Benutzerrealm
user realm

Im Realm-Eintrag der *Schema-DDL* definierter *Realm*. Er enthält u.a. die Benutzersätze.

Bezeichner
identifier

Name, den der Datenbankentwerfer für ein *Feld* vergibt, das UDS/SQL automatisch anlegt. Feldtyp und Feldlänge richtet UDS/SQL nach dem vorgegebenen Verwendungszweck des Feldes aus.

BFIM
BFIM

siehe *Before-Image*

BIB
BIB

(Base Interface Block) Standardschnittstelle zwischen UDS/SQL und jedem einzelnen Benutzer; enthält u.a. die *RECORD-AREA* (Benutzersätze wie im *Subschema* definiert).

Buffer Pools
buffer pools

siehe *System Buffer Pools* und *exklusiver Buffer Pool*

C

CALC-Key
CALC key

Schlüssel, dessen Schlüsselwerte durch eine *Hashroutine* in eine relative *Seitennummer* umgerechnet werden.

CALC-SEARCH-Key
CALC SEARCH key

Sekundärschlüssel, der als *Zugriffspfad* für *direkten Zugriff* über *Hashverfahren* realisiert wird.

CALC-Seite
CALC page

Seite eines *Hashbereichs*.

CALC-Tabelle
CALC table

Tabelle in einer direkten/indirekten *CALC-Seite*, deren Einträge auf die gespeicherten Sätze verweisen.
Sie enthält pro Zeile:

- den *CALC-Key*
- die *Satzfolgenummer*
- die Distanz zum zugehörigen *Seitenindex-Eintrag* (direkte CALC-Seite) bzw. den *Probable Position Pointer* (indirekte CALC-Seite)

CALL-DML
CALL DML

DML, die von verschiedenen Programmiersprachen (Assembler, COBOL, FORTRAN, PASCAL, PL/1) über die CALL-Schnittstelle angesprochen wird.

CHAIN
CHAIN

Speicherungsart für eine *Set-Occurrence*, bei der jeder *Satz* einen Zeiger auf seinen Nachfolger mitführt.

Character Separated Values (CSV)
Character Separated Values

Ausgabeformat, bei dem die Werte durch ein vorgegebenes Zeichen getrennt sind

CHECK-TABLE
CHECK-TABLE

Prüftabelle, die der *DDL*-Compiler bei der *Subschema-DDL*-Übersetzung erstellt und die vom COBOL-Compiler und von *CALL-DML* benutzt wird, um zu prüfen, ob die angegebenen *DML*-Anweisungen im *Anwenderprogramm* zulässig sind.
Sie befindet sich im *COSSD* bzw. im *SSITAB-Modul*.

Clone-Paar, Clone-Pubset, Clone-Session, Clone-Unit
clone pair, clone pubset, clone session, clone unit

Eine Clone-Unit ist die Kopie einer (Original-)Unit (logische Platte im BS2000) zu einem bestimmten Zeitpunkt („Point-in-Time-Kopie“). Die Komponente TimeFinder/Clone erstellt diese Kopie wahlweise als komplette Kopie oder als „Snapshot“.
Nach der Aktivierung sind Unit und Clone-Unit voneinander getrennt, Anwendungen können auf beide zugreifen.
Unit und Clone-Unit bilden zusammen ein Clone-Paar. TimeFinder/Clone verwaltet es in einer sogenannten Clone-Session.
Wenn es zu allen Units eines Pubsets Clone-Units gibt, so bilden diese Clone-Units zusammen das Clone-Pubset.
Details zu diesem Thema finden Sie im Handbuch „Einführung in die Systembetreuung“.

COBOL-DML
COBOL DML

In den COBOL-Sprachumfang integrierte *DML*.

COBOL-Laufzeitsystem
COBOL runtime system

Laufzeitsystem. Mehrfachbenutzbare Routinen, die der COBOL-Compiler (COBOL2000 bzw. COBOL85) zur Ausführung komplexer Anweisungen auswählt.

COBOL Subschema Directory (COSSD)
COBOL Subschema Directory (COSSD)

liefert dem COBOL-Compiler die Subschema-Informationen für die Übersetzung der DB-*Anwenderprogramme*.

Common Memory
common memory

Von mehreren Tasks gemeinsam benutzbarer Speicherbereich. Er besteht bei UDS/SQL immer aus dem *Common Pool* und dem *Communication Pool* und je nach Anwendungsfall aus dem SSITAB Pool (siehe *SSITAB-Modul*), wenn die *CALL-DML* verwendet wird. Beim Einsatz von UDS-D, besteht er zusätzlich noch aus dem *Distribution Pool* und dem *Transfer Pool*.

Common Pool
common pool

Kommunikationsbereich des *independent DBH* für die Verständigung der *DBH*-Module untereinander. Er enthält u.a. einen Ein-/Ausgabe-Puffer für *Seiten* (*Buffer Pool*s).

Communication Pool
communication pool

Kommunikationsbereich des *independent DBH* für *Anwenderprogramme*. Er dient u.a. zur Aufnahme der Base Interface Blocks (*BIB*).

Compilerdatenbank
compiler database

Die *Realms* und Dateien der *Datenbank*, die die UDS/SQL-Compiler benötigen. Dies sind:

- das *Database Directory (DBDIR)*
- der *Database Compiler Realm (DBCOM)*
- das *COBOL Subschema Directory (COSSD)*

COMPILER-SCHEMA
COMPILER-SCHEMA

UDS/SQL-internes *Schema* der *Compilerdatenbank*.

COMPILER-SUBSCHEMA
COMPILER-SUBSCHEMA

UDS/SQL-internes *Subschema* der *Compilerdatenbank*.

Compound Key
compound key

siehe *Schlüssel, zusammengesetzter*

Connectionmodul
connection module

siehe *Verbindungsmodul*

Consistency Record
consistency record

Verwaltungssatz mit Konsistenz-Zeitstempeln im *DBDIR*. Bei einer Änderung in einem *Realm* trägt der *DBH* im Consistency Record und im geänderten Realm Datum und Uhrzeit ein. Beim Anschließen von *Datenbanken* oder Realms an eine *Session* überprüft der DBH anhand dieser Zeitstempel, ob die Realms jeder Datenbank unter dem Konsistenzaspekt zueinander passen.

COSSD
COSSD

siehe *COBOL Subschema Directory*.

CRA
CRA

(Current Record of Area) *Satz*, der in der *Currency-Tabelle* als aktueller Satz eines bestimmten *Realm* (Area) verzeichnet ist.

CRR
CRR

(Current Record of Record) *Satz*, der in der *Currency-Tabelle* als aktueller Satz einer bestimmten *Satzart* (Record) verzeichnet ist.

CRS
CRS

(Current Record of Set) *Satz*, der in der *Currency-Tabelle* als aktueller Satz eines bestimmten *Set* verzeichnet ist.

CRU
CRU

(Current Record of Rununit) *Satz*, der in der *Currency-Tabelle* als aktueller Satz der *Verarbeitungskette* verzeichnet ist.

CSV
CSV

siehe Character Separated Values

Currency-Tabelle
currency table

Die Currency-Tabelle enthält

- die CURRENT-OF-AREA-Tabelle (Tabelle der *CRAs*),
- die CURRENT-OF-RECORD-Tabelle (Tabelle der *CRRs*),
- die CURRENT-OF-SET-Tabelle (Tabelle der *CRSs*).

CURRENT-OF-AREA-Tabelle
CURRENT OF AREA table

siehe *Currency-Tabelle*

CURRENT-OF-RECORD-Tabelle
CURRENT OF RECORD table

siehe *Currency-Tabelle*

CURRENT-OF-SET-Tabelle
CURRENT OF SET table

siehe *Currency-Tabelle*

D

DAL
DAL

(Database Administrator Language) Datenbankadministratorsprache für Kommandos zum Überwachen und Steuern einer *Session*.

Database Compiler Realm (DBCOM)
database compiler realm (DBCOM)

Speichert Einzelheiten über die *Realms*, *Sätze* und *Sets*, die der Anwender in der *Schema-DDL* und der *Subschema-DDL* definiert hat.

Database Directory (DBDIR)
database directory (DBDIR)

Enthält u.a. die *SIA*, alle *SSIAs* und Informationen über die *Zugriffsberechtigungen*.

Database Key
database key

Schlüssel, dessen Schlüsselwerte einen *Satz* in der *Datenbank* eindeutig identifizieren. Er setzt sich aus einer *Satzartnummer* und einer *Satzfolgenummer* zusammen. Die Schlüsselwerte können vom Datenbankprogrammierer vergeben oder von UDS/SQL automatisch erzeugt werden.

Database-Key-Feld
database key item

Feld vom Typ DATABASE-KEY oder DATABASE-KEY-LONG, das für die Aufnahme von *Database-Key*-Werten definiert wird.
Felder vom Typ DATABASE-KEY und Felder vom Typ DATABASE-KEY-LONG unterscheiden sich hinsichtlich der Feldlänge (4 byte / 8 byte) und des Wertebereichs.

DATABASE-KEY-Feld
DATABASE-KEY item

siehe *Database-Key-Feld*

DATABASE-KEY-LONG-Feld
DATABASE-KEY-LONG item

siehe *Database-Key-Feld*

DATABASE-STATUS
DATABASE-STATUS

5 byte langes Feld zur Anzeige des Datenbankzustands. Der Datenbankzustand besteht aus dem *Anweisungscode* und dem *Statuscode*.

Datenbank (DB)
database

Zusammengehörige Datenbestände, die mit Hilfe eines *Datenbanksystems* ausgewertet, bearbeitet und verwaltet werden.
Eine Datenbank wird durch den Datenbanknamen identifiziert.
Eine UDS/SQL-Datenbank besteht aus der *Benutzerdatenbank* und der *Compilerdatenbank*.
Zum Schutz vor Datenverlust kann parallel zur Datenbank (Originaldatenbank) eine *Schattendatenbank* betrieben werden.

Datenbankadministrator
database administrator

Person, die die *Datenbank* im laufenden Betrieb verwaltet und steuert. Der Datenbankadministrator bedient die Dienstprogramme und die Database Administrator Language (*DAL*).

Datenbankkopie
database copy

Kopie einer konsistenten *Datenbank*, die zu einem beliebigen Zeitpunkt erstellt wurde.

Datenbank-Jobvariable
database job variable

Jobvariable, in der UDS/SQL Informationen über den Zustand einer *Datenbank* hinterlegt.

Datenbankseite
database page

siehe *Seite*

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.

Datenbankzustand
database status

siehe *DATABASE-STATUS*

Datendeadlock

data deadlock

siehe *Deadlock*

Datengruppe
group item

Benennbare Zusammenfassung von *Satzelementen*.

Datenschutz
data protection (privacy)

Schutz vor unberechtigtem Zugriff auf Daten. Datenschutz wird in UDS/SQL verwirklicht durch das Schema/Subschema-Konzept und die Zugriffsrechtsprüfung. Die *Zugriffsrechte* werden mit dem Dienstprogramm BPRIVACY vergeben.

Datensicherung
data backup

Schutz vor Datenverlust bei Software- oder Hardware-Fehlern.

DBCOM
DBCOM

siehe *Database Compiler Realm*

DBDIR
DBDIR

siehe *Database Directory*

DBH
DBH

(Database Handler) Programm (bzw. Programmgruppe), das den Zugriff auf die *Datenbank (en)* einer *Session* steuert und alle dabei notwendigen Verwaltungsarbeiten übernimmt.

DBH, independent
DBH, independent

siehe *independent DBH*

DBH, Ladeparameter
DBH load parameters

siehe *Ladeparameter (DBH)*.

DBH, linked-in
DBH, linked-in

siehe *linked-in DBH*

DBH-Ende
DBH end

Beenden des *DBH* Programmlaufs. DBH-Ende kann entweder *Session-Ende* oder *Session-Abbruch* sein.

DBH-Start
DBH start

Starten des *DBH* Programmlaufs. DBH-Start kann entweder *Session-Beginn* oder *Session-Wiederanlauf* sein.

DB-Key
DB key

Siehe *Database Key*.

DB-Konfiguration
DB configuration

(database configuration) Die Menge aller *Datenbanken*, die einem *DBH* während einer *Session* momentan zugeschaltet ist. Die DB-Konfiguration kann sich im Laufe einer Session ändern, durch *DAL*-Kommandos oder durch die DBH-Fehlerbehandlung.
Eine DB-Konfiguration kann zu *Session-Beginn* auch leer sein. Mit DAL-Kommandos können Datenbanken nach Session-Beginn angeschlossen werden. Mit DAL-Kommandos können aber auch während einer Session Datenbanken ausgeschlossen werden.

DB-Status-Datei
DB status file

(database status file) Enthält Informationen über die letzten zurückgesetzten *Transaktionen*. Diese Informationen werden von UTM-S und bei verteilter Verarbeitung mit UDS-D /openUTM-D zum *Session-Wiederanlauf* benötigt.

DBTT
DBTT

(Database Key Translation Table) Tabelle, in der UDS/SQL mit Hilfe eines Database-Key-Wertes die *Seitenadresse (Act-Key)* des zugehörigen *Satzes* und der zugehörigen Tabellen findet.
Die DBTT des SSIA-RECORD besteht nur aus der DBTT-Basis. Bei allen anderen Satzarten besteht die DBTT jeweils aus einer Basistabelle (DBTT-Basis) und eventuell einer der mehreren Erweiterungstabellen (DBTT-Extents), welche durch eine Online-DBTT-Erweiterung oder durch BREORG entstehen.

DBTT-Ankerseite
DBTT anchor page

Im Realm der zugehörigen DBTT liegende Seite, in der DBTT-Basis und DBTT-Extents verwaltet werden. Möglicherweise sind mehrere untereinander verkettete DBTT-Ankerseiten zur Verwaltung der DBTT nötig.

DBTT-Basis
DBTT base

siehe *DBTT*

DBTT-Extent
DBTT extent

siehe *DBTT*

DBTT-Seite
DBTT page

Seite, die die *DBTT* oder einen Teil der DBTT einer *Satzart* enthält.

DCAM
DCAM

Teil des Datenkommunikationssystems TRANSDATA

DCAM-Anwendung
DCAM application

Kommunikationsanwendung, die die Kommunikationsmethode *DCAM* benutzt.
Eine DCAM-Anwendung bietet Kommunikationsmöglichkeit zwischen

- einer DCAM-Anwendung und Datensichtstationen.
- DCAM-Anwendungen untereinander im selben oder in verschiedenen Verarbeitungsrechnern, sowie mit *entfernten Konfigurationen*.
- einer DCAM-Anwendung und einer *openUTM*-Anwendung.

DDL
DDL

(Data Description Language) Formale Sprache zur Beschreibung der logischen Datenstruktur.

Deadlock
deadlock

Gegenseitiges Blockieren von *Transaktionen*.
Ein Deadlock kann in folgenden Situationen auftreten:

- Datendeadlock: *Transaktionen* blockieren sich gegenseitig bei *konkurrierenden Zugriffen*
- Taskdeadlock: Eine *Transaktion*, die eine Sperre hält, kann diese nicht freigeben, da keine openUTM-Task frei ist. Diese Deadlock-Situation kann nur bei UDS/SQL-openUTM-Zusammenarbeit auftreten.

Descending-Key (DESC-Key)
descending key (DESC key)

Primärschlüssel eines *Set*. Der Descending-Key legt die Reihenfolge der *Membersätze* in den *Set-Occurrences* nach absteigenden Schlüsselwerten fest.

direkter Hashbereich
direct hash area

siehe *Hashbereich*

direkter Zugriff
direct access

Zugriff auf einen *Satz* über einen Feldinhalt. UDS/SQL unterstützt den direkten Zugriff über den *Database Key* sowie über *Hashverfahren* und *mehrstufige Tabellen*.

Distribution Pool
distribution pool

Kommunikationsbereich des *independent DBH* für die Verständigung von *UDSCT*, *Servertasks*, *Anwendertasks* und *Mastertask* untereinander bezüglich UDS-D-spezifischer Daten. Im Distribution Pool liegen die *Verteiltabelle* und UDS-D-spezifische Systemtabellen.

DML
DML

(Data Manipulation Language) Sprachmittel für den Zugriff auf eine UDS/SQL-*Datenbank*.

Dummy-Teiltransaktion
dummy subtransaction

Ist eine primäre *Teiltransaktion*, die UDS-D erzeugt, wenn die erste *READY*-Anweisung einer *Transaktion* eine *entfernte Datenbank* anspricht.
Die Dummy-Teiltransaktion dient dazu, die Transaktion in der *lokalen Konfiguration* bekannt zu machen, um im Fehlerfall ein Wiederherstellen der *Datenbank* zu ermöglichen.

Duplikat-Kopf
duplicates header

Enthält allgemeine Informationen über eine *Duplikat-Tabelle* bzw. eine *Seite* einer Duplikat-Tabelle:

- die Verkettung zur nächsten und zur vorhergehenden *Überlaufseite*
- die Anzahl freier Bytes in der Seite der Duplikat-Tabelle

Duplikat-Tabelle
duplicates table

Spezielle *SEARCH-Key-Tabelle*, in der ein mehrfach auftretender Schlüsselwert nur einmal gespeichert wird.
Die Duplikat-Tabelle enthält pro Schlüsselwert

- einen Tabellenindex-Eintrag mit dem Schlüsselwert und dem Verweis auf die zugehörige Tabellenzeile
- eine Tabellenzeile (DB-Key-Liste), die auf mehrere Seiten aufgeteilt sein kann, mit den *Satzfolgenummern* der *Sätze*, die diesen Schlüsselwert enthalten

Duplikat-Tabelle, Grundstufe
duplicates table, main level

Main Level bzw. Level 0; enthält einen Tabellenindex-Eintrag und den Beginn der zugehörigen Tabellenzeile (DB-Key-Liste).

dynamischer Set
dynamic set

Set, der zeitlich begrenzt durch die Dauer der *Transaktion*, *Membersätze* von Suchfragen aufnehmen kann.

E

entfernte Datenbank
remote database

Datenbank einer *entfernten Konfiguration*.

entfernte Konfiguration
remote configuration

DB-Konfigurationen, die dem *Anwenderprogramm* nicht über /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=konfigurationsname zugeordnet werden, sondern erst bei Ablauf des Anwenderprogramms über die *Verteiltabelle*. Mit entfernten Konfigurationen verkehrt das *Verbindungsmodul* des Anwenderprogramms über die *DCAM-Anwendungen*. Entfernte Konfigurationen liegen auf dem *lokalen* oder auf einem *entfernten Verarbeitungsrechner.*

entfernter Verarbeitungsrechner
remote host

Verarbeitungsrechner, der nicht lokal ist.

entferntes Anwenderprogramm
remote application program

Anwenderprogramm, das bezüglich einer bestimmten *Konfiguration* nicht lokal ist.

ESTIMATE-REPORT
ESTIMATE-REPORT

Protokollausgabe nach dem BGSIA-Lauf. Dient dazu, die Größe der *Benutzerrealms* zu schätzen.

Event-Name
event name

Name einer Ereigniskennung.

exklusiver Buffer Pool
exclusive buffer pool

Puffer, der zusätzlich zu den *System Buffer Pools* ausschließlich für die Pufferung von *Seiten* der angegebenen *Datenbank* verwendet wird.

F

Feld
item

Kleinste benennbare Dateneinheit innerhalb einer *Satzart*. Das Feld ist definiert durch Feldtyp und Feldlänge.

Folgenummer
sequence number

siehe *ALOG-Folgenummer*

FPA
FPA

siehe *Freiplatzverwaltung*.

FPA-Basis
FPA base

siehe *Freiplatzverwaltung*.

FPA-Extent
FPA extent

siehe *Freiplatzverwaltung*.

FPA-Seite
FPA page

Seite der *Freiplatzverwaltung*.

Freiplatzverwaltung (FPA)
Free Place Administration (FPA)

Freier Platz wird sowohl auf Realm-Ebene *(FPA-Seiten)*, als auch auf Seiten- und Tabellenebene verwaltet. Die Freiplatzverwaltung der Seiten erfolgt in einer Basistabelle (FPA-Basis) und eventuell in einer oder mehreren Erweiterungstabellen (FPA-Extent), welche durch eine Online-Realmerweiterung oder durch BREORG entstehen.

Fremdschlüssel
foreign key

Satzelement, dessen Werte mit den *Primärschlüsse*l werten einer anderen Tabelle (UDS /SQL-*Satzart*) übereinstimmen. Fremdschlüssel im Sinne von UDS/SQL werden im BPSQLSIA-Protokoll in der Membersatzart einer Set-Beziehung als "REFERENCES owner-satzart" qualifiziert.

Funktionscode (FC)
function code

Verschlüsselung einer *DML*-Anweisung. Wird beim *DAL*-Kommando DISPLAY und bei UDSMON ausgegeben.

H

Hashbereich
hash area

Speicherbereich, in dem UDS/SQL Daten speichert oder wiedergewinnt aufgrund der Umrechnung von Schlüsselwerten in relative *Seitennummern*. Ein Hashbereich kann sowohl die Adressen von *Sätzen* als auch die Sätze selbst enthalten.
In einem *direkten Hashbereich* sind die Sätze selbst gespeichert, während in einem *indirekten Hashbereich* die Adressen der andernorts gespeicherten Sätze enthalten sind.

HASHLIB
HASHLIB

Modulbibliothek zur Aufnahme der *Hashroutinen* einer *Datenbank*.

Hashroutine
hash routine

Modul, das ein *Hashverfahren* ausführt.

Hashverfahren
hashing

Methode, mit der ein Schlüsselwert in eine *Seitenadresse* umgerechnet wird.

I

Identifizierung
authorization

Erkennung der Benutzergruppe.

impliziter Set
implicit set

SYSTEM-Set, den UDS/SQL bildet, wenn ein *SEARCH-Key* auf Satzartebene definiert wird.

independent DBH
independent DBH

Selbständiges Programmsystem, das den simultanen Zugriff mehrerer Anwender auf eine *Datenbank (Mono-DB-Betrieb)* oder auf mehrere Datenbanken gleichzeitig *(Multi-DB-Betrieb)* ermöglicht. Der independent DBH ist als Taskfamilie konzipiert:

- eine *Mastertask (UDSSQL)*
- eine oder mehrere *Servertasks (UDSSUB)*
- eine *Administratortask (UDSADM)*

INDEX-Search-Key
INDEX search key

Sekundärschlüssel. Er wird als *Zugriffspfad* für *direkten Zugriff* über eine *mehrstufige Tabelle* realisiert.

Indexseite
index page

Seite, in der die höchsten (niedrigsten) Schlüsselwerte der nächstniedrigen Stufe einer indizierten Tabelle gespeichert werden.

Indexstufe
index level

Hierarchiestufe einer *Indexseite*.

indirekter Hashbereich
indirect hash area

siehe *Hashbereich*

Inkonsistenz
inconsistency

Widerspruch zwischen gespeicherten Informationen.

Integrität
integrity

Fehlerfreiheit und Vollständigkeit der gespeicherten Informationen

- Objekt-Integrität (Entity Integrity)
- *referentielle Integrität* (Referential Integrity)
- Benutzer-Integrität (User Integrity)

interne Versionsnummer
internal version number

Jeder *Realm* der *Datenbank*, inklusive *DBDIR* und *DBCOM*, besitzt eine interne Versionsnummer, die die Dienstprogramme (z. B. BREORG, BALTER) bei Veränderungen des Realms um eins erhöhen. Diese interne Versionsnummer steht in der *Act-Key-0-Seite* des Realms und zusätzlich im PHYS VERSION RECORD im DBDIR.

Item
item

siehe *Feld*

K

Katalogkennung
catalog identifier

Bezeichnung der gemeinschaftlichen Platte (Public Volume Set), in der die BS2000-/UDS /SQL-Dateien gespeichert sind. Die Katalogkennung ist Bestandteil des Datenbank-/Datei-Namens und in Doppelpunkte eingeschlossen: „*:catid:*“.

KDBS
KDBS

(Compatible Database Interface) Kompatible Datenbankschnittstelle. KDBS ermöglicht, Programme auf Anwendungen von *Datenbanksystemen* verschiedener Hersteller zu übertragen.

Kennwort für die UDS/SQL-Dateien
password for UDS/SQL files

Wort, mit dem die von UDS/SQL eingerichteten Dateien geschützt sind (Standardwert: C’ UDS ’). Außerdem kann der *Datenbankadministrator* Kennwörter festlegen mit PP CATPASS oder durch MODIFY-FILE-ATTRIBUTES.

Kette
chain

siehe *CHAIN*

Kommunikationspartner
communication partners

Tasks bzw. Datensichtstationen

Komprimierung
compression

Nur belegte *Felder* eines *Satzes* werden gespeichert (siehe *SSL*-Klausel COMPRESSION).

Konfiguration
configuration

siehe *DB-Konfiguration*

Konfigurationskennung
configuration user ID

Kennung, in der der *Datenbankadministrator* den *DBH* startet.

Konfigurationsname
configuration name

Frei wählbarer Name der *Datenbankkonfiguration* einer *Session*. Aus dem Konfigurationsnamen bildet der *DBH*

- den Namen der *Session-Log-File*,
- den Namen der *DB-Status-Datei* und ihrer Sicherungskopie,
- den Namen der *RLOG-Dateien*,
- den Namen der Temporären *Realms*,
- den Namen der Session-Jobvariablen
- die *Event-Namen* des *P1-Eventing*,
- den Namen der *DCAM-Anwendung* für die Administration,
- die Namen für die *Common Pools*,
- die Namen der Dump-Dateien.

konfigurationsübergreifend
interconfiguration

Mindestens eine *entfernte Konfiguration* betreffend.

konfigurationsübergreifende Konsistenz
interconfiguration consistency

Eine *verteilte Transaktion*, die in mindestens einer *entfernten Konfiguration* geändert hat, wird so beendet, dass die Änderungen entweder auf den *Datenbanken* aller beteiligten *DB-Konfigurationen* durchgeführt werden oder auf keiner Datenbank.
Die konfigurationsübergreifende Konsistenz wird sichergestellt durch das *Zwei-Phasen-Ende-Protokoll*.

konfigurationsübergreifender Deadlock
interconfiguration deadlock

Zustand wechselseitiger Blockierungen von *verteilten Transaktionen* bei *konkurrierenden Zugriffen*.

konkurrierender Zugriff
contending access

Gleichzeitiger Zugriff auf eine *Seite* aus verschiedenen *Transaktionen*.

Konsistenz
consistency

Widerspruchsfreiheit der gespeicherten Informationen.

Konsistenz, logische
consistency, logical

Widerspruchsfreiheit der gespeicherten Daten untereinander und in Bezug auf die Realität.

Konsistenz, physische
consistency, physical

Widerspruchsfreiheit der gespeicherten Daten in Bezug auf physisch richtige Speicherung sowie vollständige und richtige *Zugriffspfade* und Beschreibungsinformationen.

Konsistenz, Speicherkonsistenz
consistency, storage

siehe *physische Konsistenz*

Konsistenzfehler
consistency error

Eine Verletzung der *physischen Konsistenz* der gespeicherten Daten.

Konsistenzpunkt
consistency point

(Zeit-)Punkt, an dem die *Datenbank* konsistent ist, d.h. alle ändernden Transaktionen sind beendet und ihre Änderungen wurden im Datenbestand durchgeführt.

Konsistenzpunkt, festgeschriebener
checkpoint

Konsistenzpunkt, bei dem die ALOG-Datei gewechselt wurde und auf den jederzeit mit Hilfe des Dienstprogramms BMEND nachgefahren werden kann

Kopie
copy

siehe *Datenbankkopie*

Kopie aktualisieren
database copy update

Datenbankkopie durch Einspielen der *After-Images* auf einen festgeschriebenen *Konsistenzpunkt* vorsetzen.

L

Ladeparameter (DBH)
load parameters (DBH)

Parameter, die der *DBH* beim Starten der *Session* anfordert. Die Parameter definieren die wesentlichen Merkmale einer Session.

Linked-in-Control-System
linked-in control system

Komponente von UDS/SQL bei *linked-in DBH*, die Steuerungsaufgaben übernimmt (entspricht dem *Subcontrol-System* bei *independent DBH*).

linked-in DBH
linked-in DBH

Modul, das in das jeweilige DB-*Anwenderprogramm* eingebunden oder nachgeladen wird und die Zugriffe auf eine *Datenbank (Mono-DB-Betrieb)* oder auf mehrere Datenbanken gleichzeitig *(Multi-DB-Betrieb)* steuert.

Liste
list

Tabelle, die die *Membersätze* einer *Set-Occurrence* enthält. Dient zum *sequentiellen* und *direkten Zugriff* auf die Membersätze.
Bei einer verteilbaren Liste können die Datenseiten, die die Membersätze enthalten (Stufe-0-Seiten), über mehrere Realms verteilt sein. Die Seiten, die die übergeordneten Tabellenstufen der verteilbaren Liste enthalten, liegen alle in einem Realm (Tabellenrealm einer verteilbaren Liste).

Logging
logging

Protokollierung über alle Änderungen in der *Datenbank*.

logische Verbindung
logical connection

Zuordnung zweier *Kommunikationspartner*, die es ihnen ermöglicht, Daten auszutauschen. *DCAM-Anwendungen* kommunizieren über logische Verbindungen.

lokale Datenbank
local database

Datenbank einer *lokalen Konfiguration*.

lokale Konfiguration
local configuration

Die *Konfiguration*, die dem *Anwenderprogramm* vor seinem Aufruf mit /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=konfigurationsname zugewiesen wurde.
Mit der lokalen Konfiguration verkehrt das Anwenderprogramm über den *Communication Pool*. Die lokale Konfiguration liegt immer im Verarbeitungsrechner des Anwenderprogramms.

lokale Transaktion
local transaction

Transaktion, die nur auf die *lokale Konfiguration* zugreift.

lokale Verteiltabelle
local distribution table

Für einen *DBH* ist die *Verteiltabelle* lokal, die in seinem *Distribution Pool* liegt.

lokaler Verarbeitungsrechner
local host

Verarbeitungsrechner, in dem das *Anwenderprogramm* liegt.

lokales Anwenderprogramm
local application program

Ein *Anwenderprogramm* ist bezüglich einer *Konfiguration* lokal, wenn es über /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=konfigurationsname an sie angeschlossen wurde.

M

Mainreference
main reference

Die Mainreference dient im *DBH* der Verwaltung der zur Bearbeitung der Aufträge einer Transaktion erforderlichen Ressourcen, einschließlich solcher für die Übertragung der Aufträge vom Anwenderprogramm zum DBH und zurück.

Mainrefnummer
mainref number

Nummer, die bei *READY* der *Transaktion* zugewiesen wird. Diese Nummer ist nur zu einem Zeitpunkt eindeutig, nach Ende der Transaktion wird sie wieder einer anderen Transaktion zugewiesen.

Maske
pattern

Bei der Definition von *Feldern* eine symbolische Darstellung aller möglichen Feldinhalte.

Maskenzeichenkette
pattern string

Zeichenfolge, die eine *Maske* definiert.

Mastertask (MT)
master task

Task des *independent DBH*, in der das Modul *UDSSQL* abläuft.
Steuert das Einleiten und Beenden einer *Session* und kommuniziert direkt oder über die *Administratortask* mit dem *Datenbankadministrator*.

mehrstufige Tabelle
multi-level table

SEARCH-KEY-Tabelle, die für jeden *Satz* der zugehörigen *Satzart* bzw. für jeden *Membersatz* der zugehörigen *Set-Occurrence* eine Zeile enthält, die aus dem Schlüsselwert des Satzes und aus dem Zeiger zum Satz besteht. Wird auch als Indextabelle bezeichnet.

Member
member

siehe *Membersatz* bzw. *Membersatzart*

Member, AUTOMATIC
member, AUTOMATIC

Ein *Satz* wird beim Speichern eingehängt.

Member, MANDATORY
member, MANDATORY

Ein *Satz* kann nicht ausgehängt werden.

Member, MANUAL
member, MANUAL

Der *Satz* wird beim Speichern nicht automatisch eingehängt.

Member, OPTIONAL
member, OPTIONAL

Der *Satz* kann ausgehängt werden.

Membersatz
member record

Untergeordneter *Satz* in einer *Set-Occurrence*.

Membersatzart
member record type

Untergeordnete *Satzart* in einem *Set*.

Mono-DB-Betrieb
mono-DB operation

Der *DBH* arbeitet mit nur einer *Datenbank* einer *Konfiguration*.

Mono-DB-Konfiguration
mono-DB configuration

Nur eine *Datenbank* ist an einer *Session* beteiligt.

Multi-DB-Betrieb
multi-DB operation

Der *DBH* arbeitet mit mehreren *Datenbanken* einer *Konfiguration*.

Multi-DB-Konfiguration
multi-DB configuration

Mehrere *Datenbanken* sind an einer *Session* beteiligt.

Multi-DB-Programm
multi-DB program

Anwenderprogramm, das auf mehrere *Datenbanken* zugreift. Die Datenbanken können zu einer *Mono-* oder *Multi-DB-Konfiguration* oder zu mehreren Mono- oder Multi-DB-Konfigurationen gehören.

Multithreading-Verfahren
multithreading

Verfahren, durch das der *DBH* die Zentraleinheit (CPU) so intensiv wie möglich nutzen kann. Im 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.

N

Netz
network

Alle über TRANSDATA gekoppelten Rechner.

netzweit eindeutig
unique throughout the network

In allen zu einem *Netz* gehörenden Rechnern eindeutig.

O

offene Transaktion
open transaction

Eine nicht mit FINISH oder mit FINISH WITH CANCEL bzw. COMMIT oder ROLLBACK abgeschlossene *Transaktion*.

OLTP
OLTP

(Online Transaction Processing) Bei einer OLTP-Anwendung greift eine sehr große Anzahl von Benutzern auf die gleichen Programme und Daten zu. Dies geschieht in der Regel unter der Steuerung eines Transaktionsmonitors (TP-Monitor)

Online-DBTT-Erweiterung
online DBTT extension

Erweiterung der Anzahl der möglichen Sätze einer Satzart im laufenden Datenbankbetrieb.
Für die Administration der Online-Erweiterbarkeit von DBTTs stehen die DAL-Kommandos ACT DBTT-INCR, DEACT DBTT-INCR, DISPLAY DBTT-INCR und EXTEND DBTT zur Verfügung.
Siehe auch *automatische DBTT-Erweiterung*.

Online-Realm-Erweiterung
online realm extension

Erweiterung von *Benutzerrealms* und *DBDIR* im laufenden Datenbankbetrieb.
Für die Administration der Online-Erweiterbarkeit von Realms stehen die DAL-Kommandos ACT INCR, DEACT INCR, DISPLAY INCR, EXTEND REALM und REACT INCR zur Verfügung.
Siehe auch *automatische Realm-Erweiterung*.

Online-Sicherung
online backup

Wenn AFIM-Logging eingeschaltet ist, kann eine Sicherung der *Datenbank* im laufenden Betrieb erstellt werden. Die Online-Sicherungsfähigkeit einer Datenbank wird mit dem Dienstprogramm BMEND festgelegt.

Operatortask (OT)
operator task

siehe *Mastertask*

openUTM
openUTM

(universal transaction monitor) Universeller Transaktionsmonitor. Er ermöglicht die einfache Erstellung und den Betrieb von Transaktionsanwendungen.

Originaldatenbank
original database

Der Begriff Originaldatenbank bezieht sich lediglich auf die Namensgebung der Datenbankdateien (*dbname.dbdatei*), nicht auf den inhaltlichen Stand der Datenbank (siehe auch *Schattendatenbank*).

Owner
owner

siehe *Ownersatz* bzw. *Ownersatzart*

Ownersatz
owner record

Übergeordneter *Satz* in einer *Set-Occurrence*.

Ownersatzart
owner record type

Übergeordnete *Satzart* in einem *Set*.

P

PETA
PETA

(Preliminary End of Transaction) Anweisung bei UDS-D und openUTM-D, die ein vorläufiges Transaktionsende herbeiführt.
Die PETA-Anweisung gehört zur ersten Phase des *Zwei-Phasen-Ende-Protokolls*, das eine *verteilte Transaktion* beendet.
Die Anweisung PETA speichert ausfallsicher in der *RLOG-Datei* des lokalen *DBH*:

- alle geänderten *Seiten*
- die Rücksetz- und Sperrinformationen
- die Namen aller beteiligten *Konfigurationen*

Diese Informationen werden bei einem eventuellen *Warmstart* benötigt.

POINTER-ARRAY
pointer array

siehe *Adressliste*

PPP
PPP

siehe *Probable Position Pointer (PPP)*.

Prepared to Commit (PTC)
prepared to commit (PTC)

Teil des *Zwei-Phasen-Ende-Protokolls*:
Zustand einer *Teiltransaktion* nach Durchführen der *PETA*-Anweisung und vor Erhalt der Nachricht, ob die gesamte *Transaktion* mit FINISH oder mit FINISH WITH CANCEL beendet wird.

primäre Teiltransaktion (PTT)
primary subtransaction

Teiltransaktion, die in der *lokalen Konfiguration* abläuft.
Die erste *READY*-Anweisung einer *Transaktion* auf eine *lokale Datenbank* eröffnet die primäre Teiltransaktion.
Falls die erste READY-Anweisung eine *entfernte Datenbank* anspricht, erzeugt UDS-D eine sogenannte *Dummy-Teiltransaktion* als primäre Teiltransaktion.

Primärschlüssel (DDL)
primary key (DDL)

Der mittels "LOCATION MODE IS CALC" definierte *Schlüssel* einer *Satzart* bzw. der mittels "ORDER IS SORTED [INDEXED]" definierte ordnungsbestimmende *Schlüssel* einer Set-Occurrence. Dient außerdem zum *Direktzugriff* auf einen *Satz* oder eine Menge von Sätzen mit gleichen Schlüsselwerten oder innerhalb eines Suchintervalls.

Primärschlüssel (SQL)
primary key (SQL)

Im weiteren Sinne (SQL) ein *Satzelement*, das einen Datensatz eindeutig identifiziert.
In UDS-SQL der im BPSQLSIA-Protokoll als "PRIMARY KEY" ausgegebene Database Key eines Ownersatzes (siehe auch *Fremdschlüssel*).
Ein einen Datensatz eindeutig identifizierendes *Satzelement* ist im BPSQLSIA-Protokoll als "UNIQUE" ausgewiesen, wenn es sich nicht um den obigen "PRIMARY KEY" handelt.

PRIVACY-AND-IQF-Schema
PRIVACY-AND-IQF SCHEMA

UDS/SQL-internes *Schema* für den Zugriffsschutz.

PRIVACY-AND-IQF-Subschema
PRIVACY-AND-IQF SUBSCHEMA

UDS/SQL-internes *Subschema* für den Zugriffsschutz.

Probable Position Pointer (PPP)
probable position pointer (PPP)

Wahrscheinliche Adresse einer *Seite*, bestehend aus *Realmnummer* und *Seitennummer*. Bei einer Lageänderung von Daten aktualisiert UDS/SQL die zugehörigen Probable Position Pointer (PPP) nicht in jedem Fall.

Prüfsätze
check records

Informationselemente zum Prüfen der Datenbank. Sie haben eine variable Länge von 20 bis 271 byte.

Pubset-Deklaration
pubset declaration

siehe *UDS/SQL-Pubset-Deklaration*

Pubset-Deklarations-Jobvariable
pubset declaration job variable

Jobvariable, in der eine *UDS/SQL-Pubset-Deklaration* vereinbart wird.

P1-Eventing
P1 eventing

Verständigung der Tasks untereinander.

Q

Quellprogramm
source program

In einer Programmiersprache formuliertes, noch nicht in die Maschinensprache übersetztes Programm.

R

READY
READY

Beginn einer *Transaktion* oder *Verarbeitungskette* bei *COBOL-DML*-Programmen.

READYC
READYC

Beginn einer *Transaktion* oder *Verarbeitungskette* bei *CALL-DML*-Programmen.

Realm
realm

Benennbare physische Untereinheit der *Datenbank*. Der Realm entspricht einer Datei. Außer den *Benutzerrealms* für die Daten gibt es die Realms *DBDIR* und *DBCOM*, die UDS /SQL selbst beansprucht.

Realm-Konfiguration
realm configuration

Die *Realms* einer *Datenbank*, die an einer *Session* beteiligt sind.

Realm-Kopie
realm copy

siehe *Datenbankkopie*

Realm-Nummer
realm reference number

Realms einer *Datenbank* werden, bei 1 beginnend, aufsteigend und lückenlos nummeriert. Die Realm-Nummer (Area-Reference) ist Bestandteil der *Seitenadresse*.

RECORD AREA
RECORD AREA

siehe *Satzbereich*

REC-REF
REC-REF

(Record Reference)
siehe *Satzartnummer*

referentielle Integrität
referential integrity

Integrität der Beziehungen zwischen Tabellen (UDS/SQL-*Satzarten*).

Rekonfiguration
reconfiguration

Neugruppierung von *Datenbanken* in einer *DB-Konfiguration* nach einem *Session-Abbruch*. Voraussetzung für eine Rekonfiguration ist, dass die *SLF* gelöscht oder inhaltlich entwertet wird.

Returncode
return code

Interner Code eines aufgerufenen Programms an das aufrufende Programm.
Returncode != 0: Fehler aufgetreten.

RLOG-Datei
RLOG file

Datei zur Ablaufsicherung. In die RLOG-Datei schreibt der *DBH* während der *Session* sowohl Daten vor ihrer Änderung (*Before-Images*) als auch Daten nach ihrer Änderung *(After-Images)*. Mit Hilfe der *RLOG-Datei* kann der DBH Änderungen nicht abgeschlossener *Transaktionen* zurücksetzen. Es gibt eine RLOG-Datei pro *Konfiguration*. Die RLOG-Datei besteht aus zwei physischen Dateien.

Rollback
rollback

Rückgängigmachen aller Änderungen einer *Transaktion*.

RSQ
RSQ

siehe *Satzfolgenummer*.

RUNUNIT-ID
RUNUNIT-ID

siehe *Transaktionskennung*

S

Satz
record

Einzelne Ausprägung einer *Satzart*. Ein Satz besteht aus je einem Feldinhalt aller am Aufbau der Satzart beteiligten *Felder* und ist die kleinste Dateneinheit, die UDS/SQL über einen eindeutigen Identifizierer, den *Database Key*, verwaltet.

Satzadresse
record address

Adresse der *Seite*, in der sich der *Satz* befindet. Siehe *Seitenadresse*.

Satzart
record type

Benennbare Zusammenfassung von *Satzelementen*.

Satzart, lineare
record type, linear

Satzart, die weder *Owner* noch *Member* eines *Set* ist (entspricht Satzarten einer konventionellen Datei).

Satzartnummer
record reference number

Satzarten werden, bei 1 beginnend, aufsteigend und lückenlos numeriert. Die Satzartnummer ist Bestandteil des *Database Key*.

Satzbereich
record area

Vom Benutzer adressierbarer Bereich der *USER-WORK-AREA (UWA)*.
Der Satzbereich enthält die *Satzarten* und die implizit definierten Felder (IMPLICITLY-DEFINED-DATA-NAMES) der Datenbank wie z.B. die AREA-ID-Felder der WITHIN-Klauseln des Schemas. Die Länge des Satzbereichs ist wesentlich durch die in ihm definierten Satzarten bestimmt.

Satzelement
record element

Feld, *Vektor* oder *Datengruppe*.

Satzfolgenummer
record sequence number

Der Datenbankprogrammierer kann die Satzfolgenummer vergeben oder UDS/SQL numeriert die *Sätze* einer *Satzart* selbst, bei 1 beginnend, aufsteigend und lückenlos in der Reihenfolge wie die Sätze gespeichert werden. Die Satzfolgenummer ist Bestandteil des *Database Key*.

Satzhierarchie
record hierarchy

Owner-/Memberbeziehung zwischen *Satzarten*:
Ownersatzart ist übergeordnet
Membersatzart ist untergeordnet.

Satz-SEARCH-Key-Tabelle
record SEARCH KEY table

SEARCH-Key-Tabelle für die Auswahl eines *Satzes* aus einer *Satzart*.

SCD
SCD

(Set Connection Data) Verknüpfungsinformation für die *Sätze* einer *Set-Occurrence*.

Schattendatenbank
backup database

Sicherung sämtlicher Dateien einer *Datenbank* jeweils unter „*dbname*.*dbdatei*.*copyname*“.
Die Schattendatenbank kann zu einem beliebigen Zeitpunkt erstellt werden und ist parallel zur Originaldatenbank im Benutzungsmodus RETRIEVAL ablauffähig.
Außerdem können die bereits abgeschlossenen *ALOG-Dateien* auf ihr parallel zur UDS /SQL-*Session* mit BMEND nachgefahren werden.

Schema
schema

Formalisierte Beschreibung der in der *Datenbank* zugelassenen Datenstrukturen. Ein UDS /SQL-Schema wird mit der *Schema-DDL* beschrieben.

Schema-DDL
Schema DDL

Formale Sprache zur Beschreibung eines *Schemas*.

Schlüssel
key

Feld, das der Datenbankprogrammierer für *Direktzugriff* auf *Sätze* benutzt und für das UDS /SQL entsprechend den Angaben im *Schema* einen optimierten *Zugriffspfad* anlegt.

Schlüssel, zusammengesetzter
key, compound

Schlüssel, der aus mehreren *Schlüsselfeldern* besteht.

Schlüsselfeld
key item

Feld, das durch Angaben im *Schema* zum *Schlüssel* erklärt wird.

Schlüsselnummer
key reference number

Schlüssel werden, bei 1 beginnend, aufsteigend und lückenlos numeriert.

Schnittstelle
interface

In der Software: Speicherbereich, den mehrere Programme zum Austausch von Daten untereinander verwenden.

SEARCH-Key
SEARCH KEY

Sekundärschlüssel. *Zugriffspfade* über Sekundärschlüssel realisiert UDS/SQL über *Hashverfahren* und *mehrstufige Tabellen*.

SEARCH-Key-Tabelle
SEARCH KEY table

Mehrstufige Tabelle, die UDS/SQL als *Zugriffspfad* über einen *Sekundärschlüssel* benutzt.

Seite
page

Physische Untereinheit von *Realms*. Seiten identifiziert UDS/SQL über eindeutige Schlüssel (*Act-Key*).
Die Länge einer Seite kann wahlweise 2048 byte, 4000 byte oder 8096 byte betragen. Innerhalb derselben Datenbank müssen alle Seiten gleich lang sein. Seiten der Länge 4000 byte oder 8096 byte sind in einen *Seitencontainer* eingebettet.

Seitenadresse
page address

Bei der Seitenadresse unterscheidet man die aktuelle Adresse einer *Seite*, den *Act-Key*, und die wahrscheinliche Adresse einer Seite, den *Probable Position Pointer (PPP)*.

Seitencontainer
page container

Seiten der Länge 4000 byte oder 8096 byte sind jeweils in einen sogenannten Seitencontainer eingebettet. Der Seitencontainer besteht aus einem 64 byte langen Header, der vor der Seite liegt, und einem 32 byte langen Trailer im Anschluss an die Seite.

Seitenindex-Eintrag
page index entry

Verweist auf die Position eines *Satzes* innerhalb einer *Seite*.

Seitenkopf
page header (page info)

Die ersten 20 byte einer *Seite* (mit Ausnahme der *FPA-Basis*-Seiten und *DBTT-Seiten* der Länge 2048 byte). Sie enthalten

- den *Act-Key* der *Seite* selbst
- die Anzahl der *Seitenindex-Einträge*
- die Länge und Position der in dieser Seite noch freien Bytes
- den Seitentyp (*ACT-Key-0-Seite*, *FPA-Seite*, *DBTT-Seite*, *DBTT-Ankerseite*, allgemeine Datenseite oder *CALC-Seite*)

Seitennummer
page number

In jedem *Realm* sind die *Seiten,* bei 0 beginnend, aufsteigend und lückenlos numeriert. Die Seitennummer ist Bestandteil der *Seitenadresse*.
Seitennummer = PAM-Seitennummer-1 bei Datenbanken mit einer Seitenlänge von 2048 byte
Seitennummer = (PAM-Seitennummer-1) / 2 bei Datenbanken mit einer Seitenlänge von 4000 byte
Seitennummer = (PAM-Seitennummer-1) / 4 bei Datenbanken mit einer Seitenlänge von 8096 byte.

sekundäre Teiltransaktionen
secondary subtransactions

Teiltransaktionen, die *entfernte Konfigurationen* ansprechen.

Sekundärschlüssel
secondary key

Jeder *Schlüssel*, der nicht *Primärschlüssel* ist; dient zum *Direktzugriff* auf einen *Satz* oder eine Menge von Sätzen mit gleichen Schlüsselwerten oder innerhalb eines Suchintervalls.

sequentieller Zugriff
sequential access

Zugriff auf einen *Satz* aufgrund seiner Position innerhalb einer vorgegebenen Satzreihenfolge.

Servertask (ST)
server task

Task des *independent DBH*, in der das Modul *UDSSUB* abläuft. Die Servertask bearbeitet die Anforderungen der DB-*Anwenderprogramme*.

Session
session

Zeitraum zwischen dem Starten und dem normalen Beenden des *DBH* (*independent/linked-in DBH*), in dem mit den *Datenbanken* der *Konfiguration* gearbeitet werden kann. Im allgemeinen Fall besteht eine Session aus einer Folge von *Session-Abschnitten* und *Session-Unterbrechungen*.

Session-Abbruch
session abort

Liegt vor, wenn der *DBH* nach erfolgreichem *Session-Beginn* abnormal beendet wird. Ursachen für einen Session-Abbruch können sein: Stromausfall, Rechnerausfall, BS2000-Störung, DBH-Fehler, %TERM.

Session-Abschnitt
session section

Beginnt mit dem Starten eines *DBH* entweder bei *Session-Beginn* oder bei *Session-Wiederanlauf* und endet mit dem normalen *Session-Ende* oder mit *Session-Abbruch*.

Session-Abschnittsnummer
session section number

Nummer, die einen Session-Abschnitt eindeutig identifiziert.

Session-Beginn
session start

Liegt vor, wenn ein *DBH* unter einem *Konfigurationsnamen* gestartet wird, für den noch keine *Session-Log-File (SLF)* mit gültigem Inhalt existiert.

Session-Ende
session end

Wird erreicht durch

- *DAL* bei *independent DBH*,
- TERM in *DML-Anwenderprogrammen* bei *linked-in DBH*,
- die *DBH*-Fehlerbehandlung.

Während einer *Session-Unterbrechung* kann das Session-Ende auch erreicht werden, indem der Anwender die *SLF* inhaltlich entwertet. Bei inkonsistenten *Datenbanken* kann die *Konsistenz* auch ohne SLF mit *Warmstart* wiederhergestellt werden.

Session-Jobvariable
session job variable

Jobvariable, in der UDS/SQL Informationen über eine Session hinterlegt.

Session-Log-File (SLF)
Session Log File (SLF)

Datei, die einer *Session* fest zugeordnet ist und die der *DBH* bei einem eventuellen *Session-Wiederanlauf* benötigt. Sie enthält Informationen über die aktuelle *DB-Konfiguration*, die Menge der aktuellen Dateikennwörter und über die aktuellen Werte der *DBH-Ladeparameter*.

Session-Unterbrechung
session interrupt

Zeitraum zwischen einem *Session-Abbruch* und dem zugehörigen *Session-Wiederanlauf*.

Session-Wiederanlauf
session restart

Start des *DBH* nach einer abgebrochenen *Session* unter gleichem *Konfigurationsnamen* und in der gleichen *Konfigurationskennung*. Mit Hilfe der *SLF* werden die *DBH-Ladeparameter* und die aktuellen Datei-Kennwörter wiederhergestellt, die bei *Session-Abbruch* vorlagen und die *Datenbanken* der damaligen *Konfiguration* werden ggf. mit *Warmstart* angeschlossen.

Set
set

Benennbare Beziehung zwischen zwei *Satzarten*.

Set, dynamischer
set, dynamic

siehe *dynamischer Set*

Set, impliziter
set, implicit

siehe *impliziter Set*

Set, singulärer
set, singular

siehe *SYSTEM-Set*

Set, Standard-
set, standard

siehe *Standard-Set*

Setnummer
set reference number

Sets werden, bei 1 beginnend, aufsteigend und lückenlos numeriert.

Set-Occurrence
set occurrence

Einzelne Ausprägung eines *Set*. Eine Set-Occurrence besteht aus genau einem *Ownersatz* und beliebig vielen ihm untergeordneten *Membersätzen*.

Set-SEARCH-Key-Tabelle
set SEARCH KEY table

SEARCH-Key-Tabelle für die Auswahl eines *Membersatzes* aus einer *Set-Occurrence*.

Shared User Buffer Pool
Shared User buffer pool

Gemeinsamer Puffer mehrerer Datenbanken, der zusätzlich zu den *System Buffer Pools* ausschließlich für die Pufferung von *Seiten* der ihm zugewiesenen *Datenbanken* verwendet wird.

SF-Pubset
SF pubset

siehe *Single-Feature-Pubset*

SIA
SIA

(Schema Information Area) Sie enthält die vollständige Datenbankbeschreibung. Der *DBH* lädt die SIA zum Arbeiten generell in den Hauptspeicher.

SIB
SIB

(SQL Interface Block) Schnittstelle zwischen UDS/SQL und SQL-Anwenderprogramm(en); enthält die SQL-Anweisung mit eventuell vorhandenen Parametern und das Anweisungsergebnis.

Single-Feature-Pubset
single feature pubset

Ein Single-Feature-Pubset (SF-Pubset) besteht aus einer oder mehreren homogenen Platten, die in den wesentlichen Eigenschaften (Plattenformat, Allokierungseinheit) übereinstimmen müssen.

SLF
SLF

siehe *Session-Log-File (SLF)*.

SM-Pubset
SM pubset

siehe *System-Managed-Pubset*

Snap-Paar, Snap-Pubset, Snap-Session, Snap-Unit
snap pair, snap pubset, snap session, snap unit

Eine Snap-Unit ist die Kopie einer (Original-)Unit (logische Platte im BS2000) zu einem bestimmten Zeitpunkt („Point-in-Time-Kopie“). Die Komponente TimeFinder/Snap erstellt diese Kopie als „Snapshot“ nach der „Copy-On-First-Write-Strategie“: Nur wenn Daten geändert werden, werden zuvor die jeweiligen Original-Daten in einen zentralen Speicherbereich (Save-Pool) des Symmetrix-Systems geschrieben. Die Snap-Unit enthält die Verweise (Track-Pointer) auf die Original-Daten. Bei unveränderten Daten zielen die Verweise auf die Unit, bei veränderten auf den Save-Pool.
Nach der Aktivierung sind Unit und Snap-Unit voneinander getrennt, Anwendungen können auf beide zugreifen.
Unit und Snap-Unit bilden zusammen ein Snap-Paar. TimeFinder/Snap verwaltet es in einer sogenannten Snap-Session.
Wenn es zu allen Units eines Pubsets Snap-Units gibt, so bilden diese Snap-Units zusammen das Snap-Pubset.
Details zu diesem Thema finden Sie im Handbuch „Einführung in die Systembetreuung“.

Sort-Key-Tabelle
sort key table

Zusätzlicher *Direktzugriff*spfad mittels des *Primärschlüssels* auf Setebene auf die *Membersätze* einer *Set-Occurrence* bei "MODE IS CHAIN" und "ORDER IS SORTED INDEXED".

spanned record
spanned record

Satz, der länger ist als eine *Seite*. Spanned records gibt es **nur UDS/SQL-intern**. Benutzersatzarten dürfen generell nicht länger sein als

- 2020 byte bei 2048 byte Seitenlänge
- 3968 byte bei 4000 byte Seitenlänge
- 8064 byte bei 8096 byte Seitenlänge

SQL
SQL

(Structured Query Language) SQL ist eine relationale Datenbanksprache, die von der ISO (International Organization for Standardization) standardisiert worden ist.

SQL-DML
SQL-DML

Data Manipulation Language von *SQL*, für die Abfrage und Änderung von Daten.

SQL-Transaktion
SQL transaction

Zusammengehörige Folge von *SQL*-Anweisungen, die UDS/SQL entweder ganz oder gar nicht bearbeitet, um die *Datenbank(en)* von einem konsistenten Zustand in einen anderen konsistenten Zustand zu überführen.

SQL-Vorgang
SQL conversation

siehe *Vorgang*

SSIA
SSIA

(Subschema Information Area) enthält alle Subschema-abhängigen Informationen, die der *Database Handler* benötigt, um für den Anwender auf die *Datenbank* innerhalb der Möglichkeiten des aufgerufenen *Subschemas* zuzugreifen.
Der *DBH* lädt die SSIA, sobald sie bei einem *READY* angesprochen wird, in den Hauptspeicher.

SSIA-RECORD
SSIA-RECORD

UDS/SQL-interne *Satzart*, die im *Database Directory (DBDIR)* liegt. *Sätze* dieser Satzart sind u.a. die Schema Information Area (*SIA*) und die Subschema Information Areas (*SSIA*).

SSITAB-Modul
SSITAB module

Vom Dienstprogramm BCALLSI erzeugtes Modul. Es stellt die Subschema-Informationen für *CALL-DML*-Programme bereit.

SSL
SSL

(Storage Structure Language) Formale Sprache zur Beschreibung der Speicherstruktur.

Standard-Set
standard set

Set, der kein *dynamischer* oder *impliziter Set* oder *SYSTEM-Set* ist.

Statuscode
status code

Nummer, die im zweiten Teil des Feldes *DATABASE-STATUS* hinterlegt wird, und die darüber informiert, welcher Sonderzustand aufgetreten ist.

String
string

Eine Reihe aufeinanderfolgender alphanumerischer Zeichen.

Subcontrol-System
subcontrol system

Komponente des *independent DBH*, die Steuerungsaufgaben übernimmt.

Subschema
subschema

Für eine bestimmte *Anwendung* erforderlicher Teil eines *Schemas*, der für eine Anwendung in begrenztem Umfang neu strukturiert werden kann. Das Subschema wird mit der *Subschema-DDL* beschrieben.

Subschema-DDL
Subschema DDL

Formale Sprache zur Beschreibung eines *Subschemas*.

Subschemamodul
subschema module

Modul, das beim Übersetzen eines *COBOL-DML*-Programms aus der Übersetzung des *Subschemas* entsteht. Es muss in das *Anwenderprogramm* eingebunden werden und enthält die *UWA* sowie die *RECORD AREA*, die gleichzeitig Teil des Base Interface Block *(BIB)* ist. Der Name des Subschemamoduls sind die ersten acht Zeichen des Subschemanamens.

Subschemasatz
subschema record

Satz laut *Subschema-DDL*.

SUB-SCHEMA SECTION
SUB-SCHEMA SECTION

Bei einem COBOL-Programm mit *DML*-Anweisungen: Abschnitt in der DATA DIVISION zur Angabe des Schemanamens und des Subschemanamens.

Subtask (ST)
subtask

siehe *Servertask*.

System Buffer Pools
system buffer pools

Ein-/Ausgabe-Puffer für Datenbankseiten (siehe *Seite*). Sie liegen im *Common Pool (independent DBH)* bzw. *DBH*-Arbeitsbereich (*linked-in DBH*). Ihre Größe bestimmen die *DBH-Ladeparameter* 2KB-BUFFER-SIZE, 4KB-BUFFER-SIZE bzw. 8KB-BUFFER-SIZE.

Systembereich
system area

Realm, der nur von UDS/SQL benötigt wird. Zu den Systembereichen einer Datenbank zählt man:

- das *Database Directory (DBDIR)*,
- den *Database Compiler Realm (DBCOM)*,
- das *COBOL Subschema Directory (COSSD)*

Systembreak-Informationen
system break information

Kennzeichen, ob die *Datenbank* konsistente oder inkonsistente Information enthält.

System-Managed-Pubset
system managed pubset

Ein System-Managed-Pubset besteht aus einem oder mehreren Volume-Sets, die wie bei einem *SF-Pubset* eine Zusammenfassung von mehreren homogenen Platten sind; die Homogenität bezieht sich auch hier auf bestimmte physikalische Eigenschaften wie z.B. Plattenformat und Allokierungseinheit.

SYSTEM-Record
SYSTEM record

siehe *Ankersatz*

SYSTEM-Set
SYSTEM set

Set, dessen *Ownersatzart* die symbolische *Satzart* SYSTEM ist.

T

Tabelle, mehrstufige
table, multi-level

siehe *mehrstufige Tabelle*

Tabelle (SQL)
table (SQL)

Eine Tabelle im *SQL*-Sinn entspricht einer UDS/SQL-*Satzart*.

Tabellenkopf
table header

Enthält allgemeine Informationen über eine Tabelle bzw. eine *Tabellenseite*:

- die Angabe über den Tabellentyp und die Stufennummer der Tabellenseite,
- die Anzahl der reservierten und der aktuellen Einträge in dieser Tabellenseite,
- die Verkettung mit weiteren Tabellenseiten der gleichen Stufe,
- den Verweis auf die zugehörige Tabellenseite der nächsthöheren Stufe und
- den Verweis auf die Seite mit der letzten Tabelle der Grundstufe (nur bei der Tabelle der höchsten Stufe).

Tabellenseite
table page

Seite, die eine Tabelle oder einen Tabellenteil enthält. Handelt es sich um eine *Tabelle*, die sich nicht über mehrere Seiten erstreckt, oder um die höchste Stufe einer mehrstufigen *Tabelle*, so ist mit „Tabellenseite“ nur das entsprechende Objekt gemeint, nicht die ganze *Seite*.

TANGRAM
TANGRAM

(Task and Group Affinity Management) Subsystem des BS2000; dieses Subsystem plant für Taskgruppen, die bei Multitask-Anwendungen auf größere gemeinsame Datenmengen zugreifen, die Zuordnung zu den Prozessoren.

Task Attribut TP
task attribute TP

Im BS2000 gibt es 4 Task Attribute: SYS, TP, DIALOG und BATCH.
Den Task Attributen sind jeweils spezielle, für das Task-Scheduling wichtige Ablaufparameter zugeordnet.
TP zeichnet sich gegenüber den anderen Task Attributen durch eine, speziell auf die Bedürfnisse des Teilhaberbetriebs optimierte Hauptspeicher-Verwaltung aus.

Taskdeadlock
task deadlock

siehe *Deadlock*

Taskkommunikation
task communication

Verständigung der *DBH*-Module untereinander. Siehe auch *Common Pool*.

Taskpriorität
task priority

Im BS2000 kann die Priorität für eine Task festgelegt werden. Diese Priorität wird bei der Initiierung und Aktivierung der Task berücksichtigt.
Es gibt variable und feste Prioritäten.Variable Prioritäten passen sich an, feste verändern sich nicht.
(UDS/SQL-Servertasks sollen mit einer festen Priorität gestartet werden, um eine gleichbleibende Performance zu erreichen).

TCUA
TCUA

(Transaction Currency Area) enthält die Currency-Informationen.

Teiltransaktion
subtransaction

In einer verteilten *Transaktion* bilden alle *Verarbeitungsketten*, die *Datenbanken* **einer** *Konfiguration* ansprechen, eine Teiltransaktion.

Transaktion (TA)
transaction

Zusammengehörige Folge von *DML*-Anweisungen, die UDS/SQL entweder ganz oder gar nicht bearbeitet, um die *Datenbank(en)* von einem konsistenten Zustand in einen anderen konsistenten Zustand zu überführen.
Bei UDS-D:
Gesamtheit aller zu einem Zeitpunkt gestarteten *Teiltransaktionen*.

Transaktion normal beenden
transaction, commiting a

Eine *Transaktion* mit FINISH beenden, d.h. alle Änderungen festschreiben, die auf den *Datenbanken* gemacht wurden.

Transaktion zurücksetzen
transaction, rolling back a

Eine *Transaktion* mit FINISH WITH CANCEL beenden, d.h. alle Änderungen rückgängig machen, die auf den *Datenbanken* gemacht wurden.

Transaktionskennung
transaction identification (TA-ID)

Vergibt der *DBH* zur Kennzeichnung einer *Transaktion*; kann mit dem *DAL*-Kommando DISPLAY erfragt werden.

Transfer Pool
transfer pool

UDS-D-spezifischer Speicherbereich, in dem der *UDSCT* die *BIBs* von *entfernten Anwenderprogrammen* empfängt.

U

UDSADM
UDSADM

Modul des *independent DBH*. Das Modul läuft in der *Administratortask* ab.

UDSHASH
UDSHASH

Vom Dienstprogramm BGSIA erzeugtes Modul mit den Namen aller *Hashroutinen*, die in der *Schema-DDL* definiert wurden.

UDSNET
UDSNET

Verteilkomponente in der *Anwendertask.*

UDSSQL
UDSSQL

Startmodul des *independent DBH*. Das Modul läuft in der *Mastertask* ab.

UDSSUB
UDSSUB

Startmodul des *independent DBH*. Das Modul läuft in der *Servertask* ab.

UDS-D-Task UDSCT
UDS-D task UDSCT

Task, die UDS/SQL für jede *Konfiguration* startet, damit sie an der verteilten Verarbeitung mit UDS-D teilnehmen kann.

UDS/SQL / openUTM-D-Konsistenz
UDS/SQL / openUTM-D consistency

Eine *Transaktion*, die sowohl *openUTM*-Daten als auch UDS/SQL-*Datenbanken* geändert hat, wird so beendet, dass entweder die openUTM-Daten und die UDS/SQL-Datenbanken geändert werden, oder keines von beiden.

UDS/SQL-Pubset-Deklaration
UDS/SQL pubset declaration

Vereinbarung in einer *Pubset-Deklarations-Jobvariable* zur Einschränkung der UDS/SQL-Pubset-Umgebung. Dadurch wird die Gefahr durch die Mehrdeutigkeit von Dateinamen verringert bzw. vermieden.

Überlaufseite
overflow page

Seite bei *Hashbereichen* und *Duplikat-Tabellen*, die diejenigen Daten aufnimmt, die nicht mehr in die Primärseite passen. Ihr Aufbau entspricht den Seiten des Hashbereichs bzw. der Duplikat-Tabelle.

Umstrukturierung
restructuring

Änderung von *Schema-DDL* oder *SSL* bei *Datenbanken*, in denen bereits Daten gespeichert sind.

USER-WORK-AREA (UWA)
USER-WORK-AREA (UWA)

Übergabebereich zur Kommunikation zwischen *Anwenderprogramm* und *DBH*.

UTM
UTM

siehe openUTM.

UWA
UWA

siehe *USER-WORK-AREA (UWA)*.

V

Vektor
vector

Feld mit Wiederholungsfaktor. Der Wiederholungsfaktor muss größer als 1 sein. Er gibt an, wieviel Duplikate des Feldes zu dem Vektor zusammengefasst werden.

Verarbeitungskette
processing chain

Folge von *DML*-Anweisungen an eine *Datenbank* innerhalb einer *Transaktion*.

Verbindungsmodul
connection module

Modul, das in jedes UDS/SQL-*Anwenderprogramm* eingebunden werden muss und die Verbindung zum *DBH* herstellt.

Versionsnummer, interne
version number, internal

siehe *interne Versionsnummer*

Verteiltabelle
distribution table

Tabelle, die UDS-D anhand der zugewiesenen Eingabedatei im *Distribution Pool* aufbaut. Mit Hilfe der Verteiltabelle entscheidet die Verteilkomponente in der *Anwendertask*, ob eine *Verarbeitungskette* lokal oder entfernt bearbeitet werden soll.
In der Verteiltabelle ist zugeordnet:
Subschema - *Datenbank*
Datenbank - *Konfiguration*
Konfiguration - Verarbeitungsrechner.

verteilte Datenbanken
distributed database

Ein logisch zusammengehörender Datenbestand, der auf mehrere UDS/SQL-Konfigurationen verteilt ist.

verteilte Transaktion
distributed transaction

Transaktion, die auf mindestens eine *entfernte Konfiguration* zugreift.
Eine Transaktion kann verteilt sein über:

- UDS-D,
- openUTM-D,
- UDS-D und openUTM-D.

Vorgang
conversation

In einer *Anwendung* mit *SQL* werden *SQL*-spezifische Verwaltungsdaten über Transaktionsgrenzen hinweg aufbewahrt. Eine solche Verwaltungseinheit wird als Vorgang bezeichnet.

W

Warmstart (einer DB)
warm start

Ein Warmstart wird von UDS/SQL durchgeführt, wenn eine inkonsistente *Datenbank* an eine *Session* angeschlossen wird. Ein Warmstart umfasst das Nachfahren der Änderungen abgeschlossener *Transaktionen*, die noch nicht auf der Datenbank festgeschrieben waren, den *Rollback* aller auf der Datenbank offenen Transaktionen und das Konsistentmachen der Datenbank. Für einen Warmstart wird die zugehörige *RLOG-Datei* benötigt und die *DB-Status-Datei*.

Wiederanlauf (von BMEND)
restart of BMEND

Fortsetzung eines abgebrochenen BMEND-Laufs.

Wiederanlauf (einer Session)
restart of a session

siehe *Session-Wiederanlauf*

Wiederholungsgruppe
repeating group

Datengruppe mit Wiederholungsfaktor. Der Wiederholungsfaktor muss größer als 1 sein. Er gibt an, wieviele Duplikate der Datengruppe zu der Wiederholungsgruppe zusammengefasst werden.

Z

Zeitquittung
time acknowledgment

Nachrichten, die die *UDS-D-Task* zum entfernten *Anwenderprogramm* sendet, um mitzuteilen, dass noch eine *DML*-Anweisung bearbeitet wird.

Zugriff, direkter
access, direct

siehe *direkter Zugriff*

Zugriff, konkurrierender
access, contending

siehe *konkurrierender Zugriff*

Zugriff, sequentieller
access, sequential

siehe *sequentieller Zugriff*

Zugriffsart
access type

Art und Weise des Zugriffs, zum Beispiel Lesen, Ändern usw.

Zugriffsberechtigte
authorized users

Festgelegte Benutzergruppen und deren Benutzer, die auf die *Datenbank* zugreifen dürfen.

Zugriffsberechtigung
access authorization

Recht einer definierten Benutzergruppe in definierter Weise auf die *Datenbank* zuzugreifen. Die Zugriffsberechtigung wird im laufenden Datenbankbetrieb mit dem Dienstprogramm ONLINE-PRIVACY bzw. im Offline-Modus mit dem Dienstprogramm BPRIVACY festgelegt.

Zugriffspfad
access path

Hilfsmittel, um eine bestimmte, durch eine Suchfrage qualifizierte Untermenge aller *Sätze* auffinden zu können, ohne die ganze *Datenbank* sequentiell absuchen zu müssen.

Zugriffsrechte
access rights

Zugriffsrechte werden durch das Dienstprogramm BPRIVACY festgelegt. Sie regeln den Zugriff auf die *Datenbank*.

Zustand PTC
PTC state

siehe *Prepared to Commit*

Zwei-Phasen-Ende-Protokoll
two-phase commit protocol

Verfahren, um eine *verteilte Transaktion*, die in mindestens einer *entfernten Konfiguration* geändert hat, so zu beenden, dass die *konfigurationsübergreifende Konsistenz* bzw. die UDS/SQL-/openUTM-D-Konsistenz gesichert ist.
Das Zwei-Phasen-Ende-Protokoll wird gesteuert:

- von der Verteilkomponente in der *Anwendertask*, wenn die *Transaktion* über UDS-D verteilt ist.
- von openUTM-D, wenn die Transaktion über openUTM-D bzw. über openUTM-D und über UDS-D verteilt ist.

11 Abkürzungen

ACS	Alias Catalog Service
Act-Key	Actual-Key
AFIM	After-Image
AP	Anwenderprogramm, Application Program
ASC	Ascending
BIB	Base Interface Block
BFIM	Before-Image
COBOL	Common Business Oriented Language
CODASYL	Conference on Data System Languages
CRA	Current Record of Area
CRR	Current Record of Record
CRS	Current Record of Set
CRU	Current Record of Rununit
COSSD	COBOL Subschema Directory
DAL	Database Administration Language
DB	Datenbank
DBCOM	Database Compiler Realm
DBDIR	Database Directory
DBH	Database Handler
DB-Key	Database Key
DBTT	Database Key Translation Table
DDL	Data Description Language
DESC	Descending
DML	Data Manipulation Language
DRV	Dual Recording by Volume
DSA	Database System Access
DSSM	Dynamische Verwaltung von Subsystemen
FC	Function Code
FPA	Free Place Administration
GS	Global Store (Globalspeicher)

HSMS	Hierarchisches Speicher Management System
ID	Identification (Kennung)
IMON	Installation Monitor
IQL	Interactive Query Language
IQS	Interactive Query System
KDBS	Kompatible Datenbank-Schnittstelle
KDCS	Kompatible Datenkommunikations-Schnittstelle
LM	Lock Manager
LMS	Library Maintenance System
MPVS	Multiple Public Volume Set
MR-NR	Mainref-Number
MT	Mastertask
OLTP	Online Transaction Processing
openUTM	Universeller Transaktionsmonitor
OT	Operatortask
PETA	Preliminary End of Transaction
PPP	Probable Position Pointer
PTC	Prepared to Commit
PTT	Primäre Teiltransaktion
PVS	Public Volume Set
REC-REF	Record-Reference
RSQ	Record-Sequence-Number (Satzfolgenummer)
SC	Subcontrol
SCD	Set Connection Data
SCI	Software Configuration Inventory
SECOLTP	Secure Online Transaction Processing
SECOS	Security Control System
SET-REF	Set-Reference
SIA	Schema Information Area
SIB	SQL Interface Block
SLF	Session-Log-File
SQL	Structured Query Language

SSD	Solid State Disk
SSIA	Subschema Information Area
SSITAB	Subschema Information Table
SSL	Storage Structure Language
ST	Servertask
STT	Sekundäre Teiltransaktion
TA	Transaction
TA-ID	Transaction-Identification
TANGRAM	Task and Group Affinity Management
TCUA	Transaction Currency Area
UDS/SQL	Universelles Datenbanksystem/Structured Query Language
UWA	User Work Area

12 Literatur

Die Handbücher finden Sie im Internet unter https://bs2manuals.ts.fujitsu.com. Handbücher, die mit einer Bestellnummer angezeigt werden, können Sie auch in gedruckter Form bestellen.

UDS/SQL (BS2000)
Anwendungen programmieren
Benutzerhandbuch

UDS/SQL (BS2000)
Datenbankbetrieb
Benutzerhandbuch

UDS/SQL (BS2000)
Entwerfen und Definieren
Benutzerhandbuch

UDS/SQL (BS2000)
Meldungen
Benutzerhandbuch

UDS/SQL (BS2000)
Sichern, Informieren und Reorganisieren
Benutzerhandbuch

UDS/SQL (BS2000)
Taschenbuch

UDS (BS2000)
Dialogsystem IQS
Benutzerhandbuch

UDS-KDBS (BS2000)
Kompatible Datenbankschnittstelle
Benutzerhandbuch

SQL für UDS/SQL
Sprachbeschreibung

BS2000 OSD/BC
Kommandos
Benutzerhandbuch

BS2000 OSD/BC
Einführung in die Systembetreuung
Benutzerhandbuch

BS2000 OSD/BC
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

BS2000 OSD/BC
Einführung in das DVS
Benutzerhandbuch

SDF (BS2000)
Dialogschnittstelle SDF
Benutzerhandbuch

SORT (BS2000)
Benutzerhandbuch

SPACEOPT (BS2000)
Optimierung und Reorganisation von Platten
Benutzerhandbuch

LMS (BS2000)
SDF-Format
Benutzerhandbuch

DSSM/SSCM
Verwaltung von Subsystemen in BS2000
Benutzerhandbuch

ARCHIVE (BS2000)
Benutzerhandbuch

DRV (BS2000)
Dual Recording by Volume
Benutzerhandbuch

HSMS / HSMS-SV (BS2000)
Hierarchisches Speicher Management System
Band 1: Funktionen, Verwaltung und Installation
Benutzerhandbuch

SECOS (BS2000)
Security Control System
Benutzerhandbuch

openNet Server (BS2000)
BCAM
Referenzhandbuch

DCAM (BS2000)
Programmschnittstellen
Beschreibung

DCAM (BS2000)
Makroaufrufe
Benutzerhandbuch

OMNIS/OMNIS-MENU (BS2000)
Funktionen und Kommandos
Benutzerhandbuch

OMNIS/OMNIS-MENU (BS2000)
Administration und Programmierung
Benutzerhandbuch

openUTM
Konzepte und Funktionen
Benutzerhandbuch

openUTM
Anwendungen programmieren mit KDCS für COBOL, C und C++
Benutzerhandbuch

openUTM
Anwendungen generieren
Benutzerhandbuch

openUTM
Anwendungen administrieren
Benutzerhandbuch

openUTM
Einsatz von openUTM-Anwendungen unter BS2000
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose (BS2000)
Benutzerhandbuch

COBOL2000 (BS2000)
COBOL-Compiler
Sprachbeschreibung

COBOL2000 (BS2000)
COBOL-Compiler
Benutzerhandbuch

COBOL85 (BS2000)
COBOL-Compiler
Beschreibung

COBOL85 (BS2000)
COBOL-Compiler
Benutzerhandbuch

CRTE (BS2000)
Common Runtime Environment
Benutzerhandbuch

DRIVE/WINDOWS (BS2000)
Programmiersystem
Benutzerhandbuch

DRIVE/WINDOWS (BS2000)
Programmiersprache
Sprachbeschreibung

DRIVE/WINDOWS (BS2000)
Lexikon der DRIVE-Anweisungen
Referenzhandbuch

DRIVE/WINDOWS (BS2000/SINIX)
Lexikon der DRIVE-SQL-Anweisungen für UDS
Referenzhandbuch

DAB (BS2000)
Disk Access Buffer
Benutzerhandbuch

XHCS (BS2000)
8-bit-Code- und Unicode-Unterstützung im BS2000
Benutzerhandbuch

Unicode im BS2000
Übersichtshandbuch

BS2000 OSD/BC
Softbooks Deutsch
CD-ROM

openSM2 (BS2000)
Software Monitor
Benutzerhandbuch

SNMP Management (BS2000)
Benutzerhandbuch