Deutsch

FUJITSU Software BS2000

UDS/SQL V2.8

Entwerfen und Definieren

Benutzerhandbuch

Ausgabe März 2016

Kritik… Anregungen… Korrekturen…

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

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

Zertifizierte Dokumentation nach DIN EN ISO 9001:2008

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

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

Copyright und Handelsmarken

Inhalt

1 Einleitung

Das **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 Seite 13.

Als Einstieg dient Ihnen das vorliegende 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 vorliegenden 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
Handbuch UDS/SQL Entwerfen und Definieren								
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	–	–	–	–

Tabelle 1: Wegweiser durch die Handbücher (Teil 1 von 3)

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
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	–	–	–

Tabelle 1: Wegweiser durch die Handbücher (Teil 2 von 3)

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
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	–	D	–	–	–	–
SQL für UDS/SQL Sprachbeschreibung	–	–	D	–	D	–	–	–

Tabelle 1: Wegweiser durch die Handbücher (Teil 3 von 3)

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 Anwender bestimmt, die die Aufgabe haben, die logische und physische Struktur der Datenbank zu entwerfen, sie mit der DDL und der SSL zu beschreiben und die Daten über die Definition geeigneter Subschemata bedarfsgerecht zur Verfügung zu stellen.
Die Sprachbeschreibungen richten sich außerdem an den Programmierer von Datenbankanwendungen und den Datenbankadministrator.
Der allgemeine Teil liefert Wissenswertes für jeden, der mit UDS/SQL arbeiten will.

1.3 Konzept des Handbuchs

Was enthält dieses Handbuch?

Im Kapitel „Allgemeines“ finden Sie Informationen zur Datenhaltung und zum Einsatz von Datenbanken. Es werden die verschiedenen Datenbankmodelle und deren Abgrenzung erklärt. Den Abschluss bildet ein allgemeiner Überblick zu UDS/SQL.

Das Kapitel „Entwurf der Datenbank“ gibt einen Überblick über Phasen des Datenbank-Entwurfs und stellt Möglichkeiten zur Verteilung der Daten vor. Es schließt sich ein Abschnitt an, der die technische Umsetzung des konzeptionellen Schemas in eine UDS/SQL-Datenbank kurz beschreibt.

Das Kapitel „Schema-DDL“ geht auf die Datendefinitionssprache (DDL) ein, mit der die logische Datenstruktur definiert wird.

Das Kapitel „SSL“ beschreibt die Speicherstruktursprache (SSL), mit der die physische Speicherung der Daten optimiert werden kann.

Das Kapitel „Definition der Benutzerschnittstelle der Datenbank“ informiert über die Möglichkeiten der Subschema-DDL und über das relationale Schema.

Die Kapitel „Aufbau der Seiten“ und „Aufbau der Sätze und Tabellen“ liefern Informationen für den Anwender, der an speziellen Details interessiert ist.

Das Kapitel „Nachschlageteil“ fasst die Syntax von Schema-DDL, SSL und Subschema-DDL zusammen.

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 *http://manuals.ts.fujitsu.com* zur Verfügung. Alternativ finden Sie Readme-Dateien auch auf der Softbook-DVD.

Informationen unter BS2000

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

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

Diese Datei enthält eine kurze Information zur Readme-Datei in deutscher oder englischer Sprache (<lang>=D/E). Die Information können Sie am Bildschirm mit dem Kommando `/SHOW-FILE` oder mit einem Editor ansehen.

Das Kommando `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` zeigt, unter welcher Benutzerkennung die Dateien des Produkts abgelegt sind.

Ergänzende Produkt-Informationen

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

1.4 Änderungen gegenüber den Vorgänger-Handbüchern

In der folgenden Tabelle 2 sind die wichtigsten Änderungen der Version UDS/SQL V2.8 gegenüber der Version V2.7 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
UDSMON-Utility: Verbesserungen bei Transaktionszeit und DB COUNTERS		
Für die Ausgabe auf Terminal und Drucker: In der UDS/SQL-Monitor-Maske wird COUNTER, die Einheit für die Ausgabe der AVG TRANSACTION TIME, präzisiert auf Sekunden und Millisekunden für das Monitoring von kurzen Transaktionen.	DBB	11
Neues DISPLAY DBCOUNTERS-Kommando in UDSMON für die Ausgabe von Datenbankzählern.	DBB	11
BSTATUS-Utility: Begrenzung des TABLE STATISTICS FOR OWNER IN SET		
Verbesserte DISPLAY TABLE FOR OWNER-Anweisung, um eine Begrenzung von TABLE STATISTICS FOR OWNER IN SET auf Sätze von bestimmten Ownern oder Satzbereiche zu ermöglichen.	SIR	6
Neue Meldungen des Dienstprogramms BSTATUS	MEL	3

Tabelle 2: Änderungen in V2.8 gegenüber V2.7

Thema	Handbuch	Kapitel
Neue Meldung des Dienstprogramms BPRECORD 2553 falls der Wert 0 als Startwert in RSQ range definiert wird.	MEL	3
Datenbankbetrieb: Die Anzahl der DML-Anweisungen und die Anzahl der Ein- und Ausgaben werden pro Datenbank gezählt.	DBB MEL	4 2
BOUTLOAD-Utility: Ausgabe im CSV-Format	AUF MEL	5 3
COPY-RECORD-Anweisung: Neuer Operand CSV-OUTPUT	AUF	5
Neues Ausgabedateiformat CSV	AUF	5
Online-Utility – Probable Position Pointers (PPP) reorganisieren		
Neue DML REORGPPP - PPPs reorganisieren	SIR	8
Neue SDF-Anweisungen: SET-REORGANIZE-PPP-PARAMETERS, SHOW-REORGANIZE-PPP-PARAMETERS	SIR	8
Neue Prozedur-Anweisung REORGPPP	SIR	8
Neue vordefinierte Variablen: REORG-PPP-CURRENT, REORG-PPP-LOCKED, REORG-PPP-PAGES	SIR	8
Neue vordefinierte Standardprozedur *STDREPPP	SIR	8
Neues Beispiel „Reorganisieren von PPPs“	SIR	8
Neue Statuscodes mit Fortschrittshinweisen der Online-Utility REORGPPP und neue Fehlercodes	ANW	10

Tabelle 2: Änderungen in V2.8 gegenüber V2.7

Allgemeine Änderung

Die bisherige Bezeichnung BS2000/OSD-BC des BS2000-Grundausbaus ändert sich und lautet ab Version V10.0: BS2000 OSD/BC.

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

	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. 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*]

Tabelle 3: Zeichen der Metasprache (Teil 1 von 2)

Sprachelement	Erklärung	Beispiel
... 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 (Teil 2 von 2)

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

2 Allgemeines

In diesem Kapitel finden Sie allgemeine Informationen zur Datenhaltung und zum Einsatz von Datenbanken. Es werden die verschiedenen Datenbankmodelle und deren Abgrenzung erklärt. Den Abschluss bildet ein allgemeiner Überblick zu UDS/SQL.

2.1 Datenorganisation im Wandel

Mit der Nutzung der Datenverarbeitung auf allen Ebenen der Unternehmenshierarchie sind die qualitativen und quantitativen Anforderungen an Datenhaltung und Datenorganisation stark gestiegen. Sie sind darüber hinaus einem ständigen Wandel unterworfen.

Die Gründe für diesen Wandel sind:

– rapides Anwachsen der Datenmengen
– Client-/Server-Anwendungen
– Übergang von der Daten- zur Informationsverarbeitung
– Einfluss neuer Kommunikationsmedien und Techniken
– zunehmende Dezentralisierungstendenzen

Die Erkenntnis, dass Daten als eigenständiger betrieblicher Produktionsfaktor neben den klassischen Betriebsmitteln Kapital/Finanzen, Personal und Anlagen von großer Bedeutung sind, hat sich heute weitgehend durchgesetzt.

Wie und durch wen die Daten auf den verschiedenen Ebenen der Unternehmenshierarchie genutzt werden, ist im folgenden Bild 1 in Form einer Informationspyramide dargestellt.

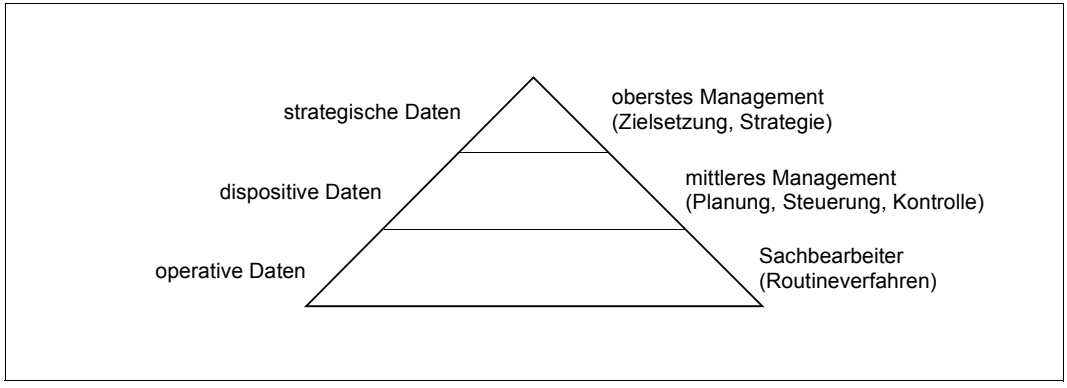


Bild 1: Durchgängigkeit der Daten und Informationen in einem Unternehmen

Die Datenbank ist die Grundlage für die physische Datenhaltung und die logische Datenorganisation eines Unternehmens. Sie steht im Mittelpunkt des betrieblichen Informationsprozesses.

Gründe für den Einsatz von Datenbanken

Die Überlegungen, ein Datenbanksystem einzusetzen, entstehen häufig daraus, dass umfangreiche Datenmengen und eine steigende Zahl von Anwendungen mit konventionellen Dateien nicht mehr zu verarbeiten sind.

Voneinander isolierte Datenbestände, die sich häufig überlappen (Datenredundanz) und in verschiedene Dateien aufgeteilt sind, lassen sich nur mit großem Aufwand aktuell halten und sind schwer zu hantieren.

Die Verfahren, die in Datenbanksystemen enthalten sind, schaffen hier Abhilfe. Sie führen zu einer stabilen, überlappungsfreien und erweiterbaren Datenorganisation und enthalten gleichzeitig komfortable und performante Möglichkeiten der Datenwiedergewinnung und -manipulation.

Werden die Daten in einem Datenbanksystem zusammengeführt und dort gemeinsam verwaltet, ist auch die Konsistenz gewährleistet: Alle Daten haben zu einem gegebenen Zeitpunkt den gleichen Änderungsstand. Dies auch dann, wenn zahlreiche Anwendungen auf diese Daten gleichzeitig zugreifen.

Datenbanksysteme schützen auch die Daten vor unbefugtem Zugriff. Sie erlauben, den Datenbestand stets verfügbar zu haben und fehlerhafte oder zerstörte Daten zu rekonstruieren.

Hohen Anforderungen an das Zeitverhalten sind Datenbanksysteme ebenfalls gewachsen, indem sie geeignete Techniken verwenden, um beispielsweise die Ein- und Ausgabegeräte der DV-Anlage optimal zu bedienen.

Der Einsatz eines Datenbanksystems bringt spürbare Entlastung der Kosten, vor allem in der Anwendungsentwicklung.

Die Kostenersparnis wird aus folgenden Gründen erzielt:

– Alle Beteiligten können auf eine einheitliche und anwendungsneutrale Datenbasis zurückgreifen.

– Es können vorgefertigte Funktionen benutzt werden.

– Eine langfristig stabile Datenorganisation löst viele Probleme.

Folgekosten für Wartung und Pflege lassen sich aus folgenden Gründen verringern:

– Daten und Programme sind sauber voneinander getrennt.

– Logische Datenorganisation und physische Speicherungsform sind voneinander unabhängig.

2.2 Datenmodelle

Das Datenbanksystem UDS/SQL unterstützt das Netzwerk-Modell (CODASYL-Modell) und das Relationenmodell. Es vereint die Prinzipien und Möglichkeiten des CODASYL-Modells und des Relationenmodells in **einem** System. Damit ist UDS/SQL eine Realisierung des so genannten Koexistenzmodells.
Die folgenden Abschnitte beschreiben kurz das CODASYL-Modell, das Relationenmodell und die Abgrenzung der beiden Modelle sowie das Koexistenzmodell.

2.2.1 CODASYL-Modell

Die Verträglichkeit von Datenbanksystemen zueinander ist eine wesentliche Forderung der Benutzer. Starke Benutzervereinigungen setzten sich deshalb schon frühzeitig die Normierung der Datenbanksysteme zum Ziel.
Die Conference on Data System Languages (CODASYL) hat weitgehend anerkannte Normierungsempfehlungen für Datenbanksysteme erarbeitet. Diese Vereinigung hatte bereits mit der Programmiersprache COBOL einen wesentlichen Beitrag zur Portabilität von DV-Anwendungen geleistet.

CODASYL wurde von amerikanischen Herstellern und Benutzern, vor allem aber unter Beteiligung der amerikanischen Regierung, im Jahre 1959 gegründet. Seit 1965 befasst sich dieses Gremium auch mit der Thematik der Datenorganisation und Datenbanken. Die grundlegenden Ergebnisse der Arbeitsgruppen wurden veröffentlicht. Sie beschreiben ein Datenbankkonzept, das von Bericht zu Bericht ständig verbessert wurde.

Das CODASYL-Modell war die Implementierungsbasis für das Datenbanksystem UDS/SQL.

Das CODASYL-Modell sieht vor, dass eine Datenbank neben den Datensätzen auch die Beziehungen dieser Datensätze untereinander enthält. Deshalb bezeichnet man es auch als Netzwerk-Modell.

Das folgende Bild 2 zeigt an einem Beispiel eine vernetzte Datenstruktur. Sie ist durch Kästchen und Pfeile dargestellt. Ein Kästchen symbolisiert eine Satzart. Im Beispiel sind die Sätze, die Lieferanten beschreiben, in der Satzart LIEFERANT zusammengefasst. Alle Sätze einer Satzart haben den gleichen Aufbau. Die Satzart LIEFERANT enthält für jeden Lieferanten einen Satz, der den Namen und den Wohnort eines Lieferanten beschreibt. NAME und WOHNORT sind Satzelementnamen.

Eine weitere Satzart in Bild 2 ist die Satzart ARTIKEL. Die Satzarten ARTIKEL und LIEFERANT sind durch einen Pfeil verbunden. Der Pfeil stellt dar, dass eine Beziehung zwischen den Satzarten existiert. Derartige Beziehungen werden Set-Beziehung oder kurz Set genannt.

Der Set zwischen den Satzarten LIEFERANT und ARTIKEL hat den Namen LIEFERT. Im Set LIEFERT ist LIEFERANT die Ownersatzart und ARTIKEL die Membersatzart. Einem Satz der Ownersatzart sind in der Regel mehrere Sätze der Membersatzart zugeordnet. Im Beispiel liefert der Lieferant Schmidt die Artikel Lebkuchen und Marzipan.

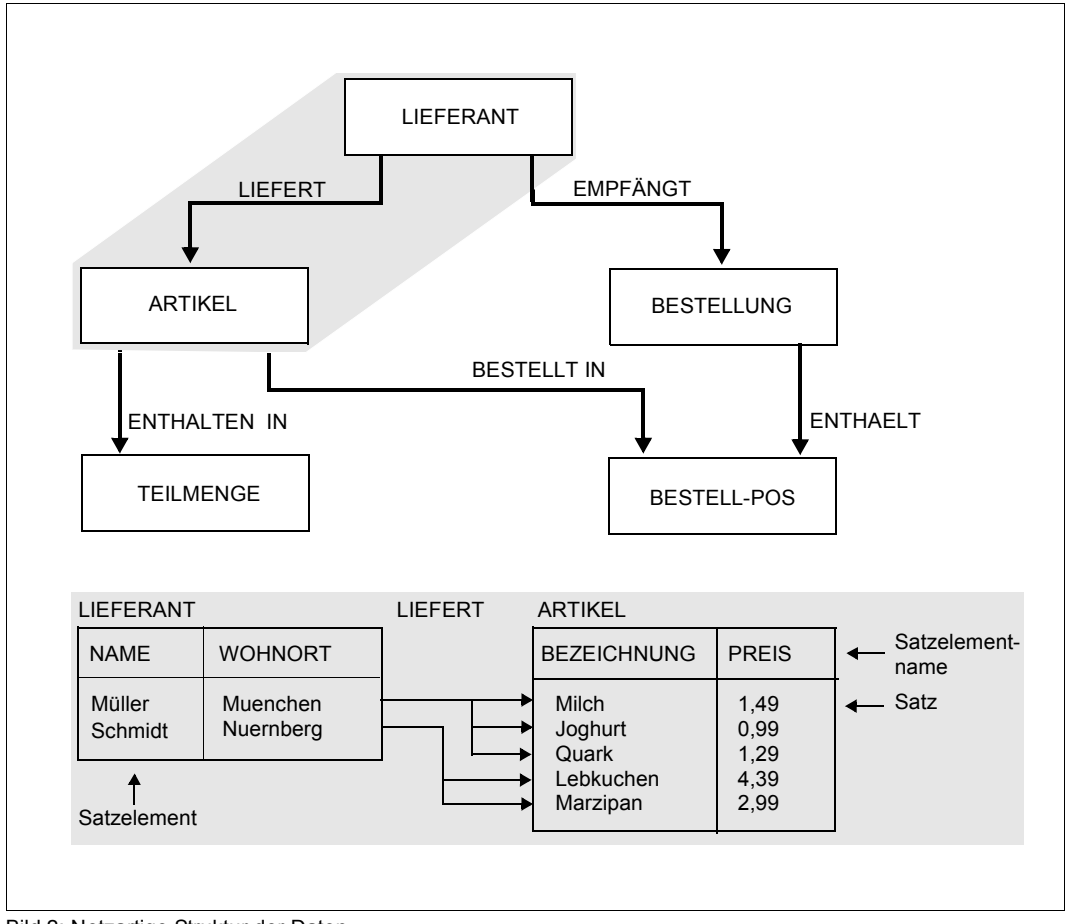


Bild 2: Netzartige Struktur der Daten

Bei einer netzwerkartigen Struktur kann eine Satzart Member in mehreren verschiedenen Set-Beziehungen sein, also auch mehrere Owner haben. Ein Datensatz kann dabei pro Set-Beziehung nur einen Owner haben.

Beziehungen zwischen Satzarten und referentielle Integrität

In Bild 2 drückt der Set LIEFERT aus, dass ein Satz der Satzart ARTIKEL kein isolierter Satz ist, sondern dass er einem Satz der Satzart LIEFERANT zugeordnet ist. Innerhalb von UDS/SQL kann im Rahmen einer Set-Definition vereinbart werden, dass es keinen Artikel geben darf ohne einen diesem Artikel zugeordneten Lieferanten. Eine derartige Bedingung ist ein Beispiel für referentielle Integrität.
Referentielle Integrität gewährleistet z.B. auch, dass die Beziehungen zwischen Auftrag und Kunde konsistent sind; in diesem Fall muss es zu jedem Auftrag einen Kunden geben. Die Einhaltung der referentiellen Integrität bedeutet hier, dass kein Auftrag aufgenommen werden kann, wenn es keinen Kunden dazu gibt, und dass kein Kunde gelöscht werden kann, solange noch Aufträge von ihm existieren.

Sprachkomponenten des CODASYL-Berichts

Der CODASYL-Bericht beschreibt drei wichtige Sprachkomponenten:

1. Die Schema-DDL (Data Description Language) beschreibt die logische Struktur der Datenbank. Es ist möglich, beliebig vernetzte Datenstrukturen zu definieren.
2. Die Subschema-DDL beschreibt benutzerspezifische Sichten der Datenbank.
   Ein Subschema ist ein Teil eines Schemas. Je Schema kann es mehrere Subschemata geben. Diese Subschemata können sich überschneiden, d.h. es kann eine Satzart in zwei oder mehreren Subschemata geben. Ein Subschema kann die gesamte Datenbank umfassen, es kann sich aber auch auf eine einzelne Satzart beschränken.

   Das Schema-Subschema-Konzept ist ein wesentlicher Baustein des Datenschutzes: Jeder Benutzer kann nur innerhalb seines Subschemas Operationen ausführen. Der übrige Teil der Datenbank ist für ihn nicht zugänglich.
3. Die COBOL-DML (Data Manipulation Language) ist eine in sich geschlossene Sprache, um auf Datenbanken zuzugreifen. Sowohl der Funktionsumfang als auch die Einbettung der DML in die Programmiersprache COBOL sind genau definiert. Die wesentlichen Sprachelemente dienen zum Steuern der Verarbeitung und zum Navigieren, Lesen und Ändern in der Datenbank.

2.2.2 Relationenmodell

Das Relationenmodell basiert auf den theoretischen Arbeiten von E.F. Codd. Dieser hat die Datenorganisation und Datenmanipulation von Datenbanksystemen mit den Mitteln der Relationenalgebra beschrieben. Er verwendete exakt festgelegte Begriffe, um das mathematische Modell seiner Relationentheorie darzustellen.

Anhand der folgenden Struktur einer kleinen Datenbank werden die Begriffe und Konzepte des Relationenmodells erklärt:

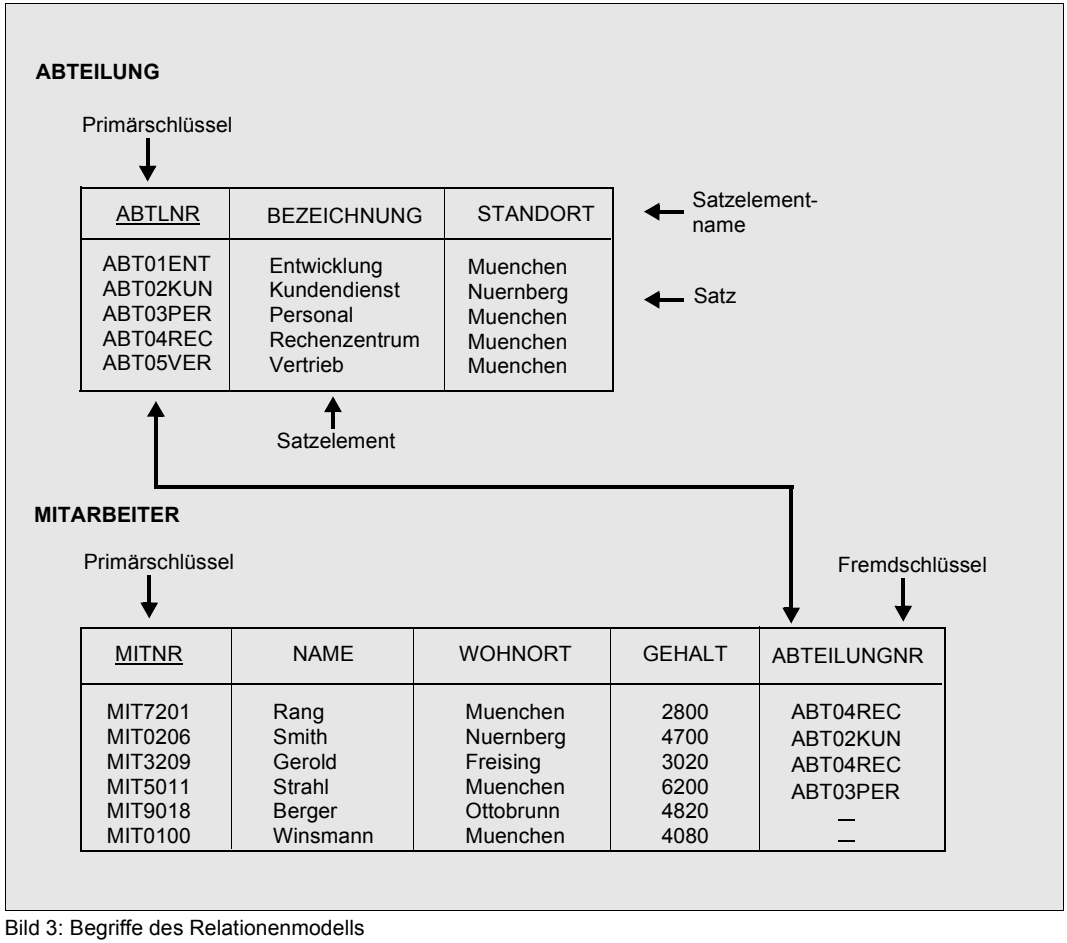


Bild 3: Begriffe des Relationenmodells

Die folgende Gegenüberstellung zeigt die in diesem Abschnitt verwendeten Begriffe und die formalen Begriffe, wie sie Codd definiert hat:

verwendeter Begriff	**formaler Begriff (Codd)**
Tabelle	Relation
Satz	Tupel
Spalte, Satzelement	Attribut
Wert	Attributwert
Wertebereich	Domäne

Im Relationenmodell werden die Daten in Form von Tabellen verwaltet und bearbeitet. Es gibt verschiedene Arten von Tabellen:

– Basistabellen
– Ergebnistabellen
– Views

Basistabelle

Basistabellen sind Tabellen, die fest in der Datenbank definiert sind. Im Bild 3 enthält die Datenbank zwei Basistabellen mit den Tabellennamen ABTEILUNG und MITARBEITER.

Eine Tabelle besteht aus Zeilen und Spalten. Die Basistabelle im Relationenmodell ist mit der Satzart im CODASYL-Modell vergleichbar. Eine Zeile einer Tabelle bezeichnet man als Satz oder Datensatz. Ein Satz der Tabelle MITARBEITER enthält alle Informationen zu einem Angestellten. Für „Spalte“ wird im Folgenden der Begriff „Satzelement“ verwendet.

Ergebnistabelle

Eine Abfrage der Datenbank führt wieder zu einer Tabelle. Diese Ergebnistabelle enthält die gefundenen Treffersätze.

View

Ein View ist ein Ausschnitt aus einer Datenbank. Ein View kann Satzelemente und Datensätze einer oder mehrerer Basistabellen enthalten. Ein View kann mit SQL-Anweisungen wie eine Basistabelle angesprochen werden. Durch die Verwendung von Views kann eine wesentliche Vereinfachung der Datenbankschnittstelle und ein hohes Maß an logischer Datenunabhängigkeit erreicht werden.

Beziehungen zwischen Tabellen

Bei der relationalen Datenorganisation wird die Beziehung von Satzarten untereinander durch bestimmte übereinstimmende Inhalte von Satzelementen realisiert. Eine Tabelle im Relationenmodell entspricht somit einer Satzart im CODASYL-Modell. Um eine Tabelle über Satzelementinhalte mit einer anderen zu verknüpfen, wird der identifizierende Primärschlüssel der ersten Tabelle als (redundanter) Fremdschlüssel in der zweiten Tabelle mit abgespeichert (siehe Bild 3, Tabellen ABTEILUNG und MITARBEITER).

Ein Fremdschlüssel muss mit den Werten eines Primärschlüssels einer anderen Tabelle übereinstimmen. Tabellen werden logisch durch Fremdschlüssel verbunden. Der Anwender definiert die Fremdschlüssel selbst.

Die Tabelle ist die einzige fixierte Datenstruktur, die das Relationenmodell kennt. Die logischen Beziehungen zwischen zwei Tabellen werden von den meisten relationalen Datenbanksystemen weder überwacht noch gepflegt. Der Anwender stellt solche Beziehungen erst zum Zeitpunkt der Datenbank-Bearbeitung über Werte her. Die Verknüpfung von Informationen erfolgt also inhaltsbezogen auf logischer Ebene. Der Anwender ist nicht eingeschränkt durch Strukturen, die vom System vorgegeben sind.

Datenmanipulation und -wiedergewinnung in relationalen Datenbanken

Das Relationenmodell beschreibt grundsätzliche, an die Mengenoperationen angelehnte Datenbankfunktionen. Somit ist neben dem Prinzip, Daten in Form von Tabellen zu verarbeiten, ein wesentlicher Bestandteil des Relationenmodells ein definierter Vorrat an Grundoperationen. Mit diesen ist es möglich, Informationen aus relationalen Datenbanken zu gewinnen.
Das Relationenmodell definiert keine Sprache zur Datenmanipulation und Datenwiedergewinnung.

Wesentliche Merkmale von Datenmanipulationssprachen (DML) in relationalen Systemen sind die mengenorientierten Operationen Projektion, Selektion und Join sowie das Fehlen vordefinierter Strukturen. Die relationalen Datenmanipulationssprachen beschreiben somit nicht den Weg, der zum gewünschten Ergebnis führt, sondern das Ergebnis selbst; d.h. der Benutzer formuliert nicht mehr **wie** das gewünschte Ergebnis in der Datenbank ermittelt werden soll, sondern nur noch **was** er sehen möchte.

Die Abdeckung des Relationenmodells innerhalb von UDS/SQL wird durch die Bereitstellung von SQL (Structured Query Language) erreicht, eine Sprache für relationale Datenbanksysteme.

SQL - eine einheitliche Sprache für relationale Datenbanksysteme

Parallel zur Entwicklung der Theorie der relationalen Datenbanken durch Codd und andere wurde an der Benutzerschnittstelle für derartige Systeme gearbeitet. Die ersten Ergebnisse wurden im Prototyp „System R“ vorgestellt und in den darauf folgenden Jahren ständig ergänzt und angepasst. Seit Anfang der 80er-Jahre sind erste kommerzielle Implementierungen dieser Sprachschnittstelle unter dem Namen SQL auf dem Markt. Mit dem Ziel, eine Vereinheitlichung dieser Sprache zu erreichen, befassen sich seit 1982 maßgebliche Standardisierungsorganisationen wie ANSI und ISO mit einer Normung der SQL.
1987 wurde unter aktiver Mitwirkung von Siemens der ISO-Standard 9075:1989(E) für SQL verabschiedet.
Die Mächtigkeit der SQL-DML erlaubt dem Anwender, alle wesentlichen Operationen auf der Datenbank auszuführen. Es können Daten in Tabellen abgefragt, eingefügt, verändert und gelöscht werden. Tabellen können miteinander verknüpft und Ergebnismengen mehrerer Suchfragen zusammengefasst werden. Grundsätzlich bewirkt die DML eine mengenorientierte Verarbeitung. So ist das Ergebnis einer Selektion wieder eine Tabelle, die der Anwender dann satz- bzw. zeilenweise abarbeiten kann. Auch Änderungsanweisungen verändern alle Sätze, die den angegebenen Bedingungen genügen. Gerade diese mengenorientierte Arbeitsweise ist ein prinzipieller Unterschied der SQL gegenüber Datenbanksprachen für nichtrelationale Datenbanksysteme.

Hier liegt auch ein wesentlicher Vorteil für den Anwender: Mit einer einzigen Anweisung wird eine Vielzahl von Aktionen auf der Datenbank ermöglicht, die im Falle einer satzorientierten Verarbeitung alle einzeln angestoßen werden müssen.

Zusammenfassend lassen sich folgende Merkmale der SQL feststellen:

– SQL basiert auf den Grundlagen des relationalen Datenmodells nach E.F.Codd und vereinigt deshalb alle seine Vorzüge: Mächtige Datenmanipulation mit deskriptiven mengenverarbeitenden Anweisungen und einfache Datenorganisation, deren Änderung auch die Programme nicht berührt.

– Durch die Verwendung der genormten SQL-Schnittstelle wird die Portabilität von DB-Anwendungen gefördert und eine weitgehende Unabhängigkeit von speziellen Datenbankschnittstellen erreicht (Erleichterung der Know-how-Beschaffung).

– SQL ist eine leicht erlernbare, einheitliche Sprache.

Jede dieser Eigenschaften führt zu einer Produktivitätssteigerung bei der Anwendungsprogrammierung und verbessert die Wirtschaftlichkeit des Datenbankeinsatzes insgesamt. Die offensichtlichen Vorteile von SQL haben dazu geführt, dass fast alle Hersteller diese Schnittstelle anbieten bzw. ankündigen.

Ein eigenes Handbuch (siehe Handbuch „SQL für UDS/SQL“) informiert eingehend über grundsätzliche Begriffe und Konzepte von SQL und beschreibt den SQL-Sprachumfang, den UDS/SQL unterstützt.

2.2.3 Abgrenzung der Datenmodelle

Die Datenmodelle gegeneinander hinsichtlich ihrer qualitativen Aussage abzugrenzen, ist schwierig und nur möglich aus der Sicht des Anwendungsgebietes. Vor allem eine Gewichtung von Vor- und Nachteilen kann nur für einen bestimmten Einsatzfall vorgenommen werden.

Grundsätzlich können jedoch alle Datenstrukturen mit jedem der beiden Modelle abgebildet werden.

Vorteile des Relationenmodells

Die Vorteile des Relationenmodells liegen eindeutig bei der größeren **Flexibilität** hinsichtlich der Datenstrukturen.

Da die Anwenderprogramme nicht von den Datenstrukturen abhängen, sind die Datenstrukturen in vielen Fällen veränderbar, ohne dass die Anwendung betroffen ist. Durch den Einsatz von Views können dem Anwender die darunter liegenden Datenstrukturen völlig verborgen werden.

Vorteile des CODASYL-Modells

Ein wesentlicher Vorteil des CODASYL-Modells liegt in der **Performanz** der Anwendung.

Das CODASYL-Modell sieht vor, neben den Satzarten bzw.Tabellen, auch deren Beziehungen untereinander explizit zu definieren. Die Beziehungen zwischen den Satzarten müssen daher auch der Anwendung bekannt sein. Dadurch, dass die Anwendung auf den definierten Datenstrukturen aufbaut, ist sie optimal auf diese abgestimmt. Diese Anpassung der Anwendung an die dem Datenbanksystem bekannten Datenstrukturen kommt der Performanz der Anwendung sehr zugute.
Der Einfluss der Datenstrukturen auf die Anwendung geht natürlich zu Lasten der Flexibilität. Änderungen der Datenstrukturen wirken sich in vielen Fällen auf die Anwendung aus.

Ein weiterer Vorteil des CODASYL-Modells liegt bei der **Überwachung der referentiellen Integrität**.

Ein Datenbanksystem, das nach dem CODASYL-Modell implementiert wurde, sorgt automatisch dafür, dass die definierten logischen Beziehungen zwischen den Satzarten nicht verletzt werden.

2.2.4 Koexistenz von CODASYL-Modell und Relationenmodell

Mit dem Einsatz von UDS/SQL ist keine Entscheidung für das CODASYL-Modell und gegen das Relationenmodell gefallen. UDS/SQL unterstützt **beide** Modelle in einem Datenbanksystem. Man spricht vom so genannten Koexistenzmodell. SQL- und CODASYL-Anwendungen können gleichzeitig mit demselben Datenbestand arbeiten.

Mit dem Koexistenzmodell stehen dem Anwender auch die Vorteile beider Datenmodelle zur Verfügung:

– hohe Flexibilität hinsichtlich der Datenstrukturen bei SQL-Anwendungen z.B. durch den Einsatz von Views

– höchste Performanz für CODASYL-Anwendungen

– Überwachung der referentiellen Integrität bei CODASYL- **und** SQL-Anwendungen, sofern Set-Beziehungen definiert wurden

UDS/SQL lässt zwei Formen der logischen Datenorganisation zu:

– CODASYL-Datenorganisation mit Set-Beziehungen zwischen den Satzarten

– Relationale Datenorganisation, bei der Satzarten nur über deren Satzelementinhalte verknüpft werden

Die SQL-Schnittstelle auf CODASYL-Datenstrukturen wird dadurch unterstützt, dass mit Hilfe des Dienstprogramms BPSQLSIA eine relationale Sicht für praktisch alle CODASYL-Strukturen generiert wird (siehe Handbuch „Sichern, Informieren und Reorganisieren“). Dies ist deshalb notwendig, weil in den SQL-Anweisungen explizit Datenelemente verwendet werden müssen, die es in der CODASYL-Datenbankbeschreibung nicht gibt: in der relationalen Datenbankbeschreibung werden Primärschlüssel zu allen Ownersatzarten hinzugefügt und Fremdschlüssel bei Membersatzarten ergänzt.

Bei der Generierung der relationalen Datenbeschreibung durch BPSQLSIA wird die Datenbank physisch nicht verändert. Die zusätzlichen Datenelemente (Primärschlüssel und Fremdschlüssel) sind nur **logisch** vorhanden. Als Ergebnis der Generierung erhält der SQL-Anwendungsprogrammierer auf einer Druckerliste alle notwendigen Informationen (z.B. Tabellennamen, Satzelementnamen, Satzelementbeschreibungen usw.), um mit SQL eine CODASYL-Datenbank zu bearbeiten. Der SQL-Anwendungsprogrammierer kann allein mit dem so genannten relationalen Schema arbeiten. Das CODASYL-Schema bleibt jedoch unverändert erhalten und kann von den CODASYL-Anwendungen weiter benutzt werden.

Bild 4 zeigt die verschiedenen Benutzersichten auf eine UDS/SQL-Datenbank, die sich daraus ergeben:

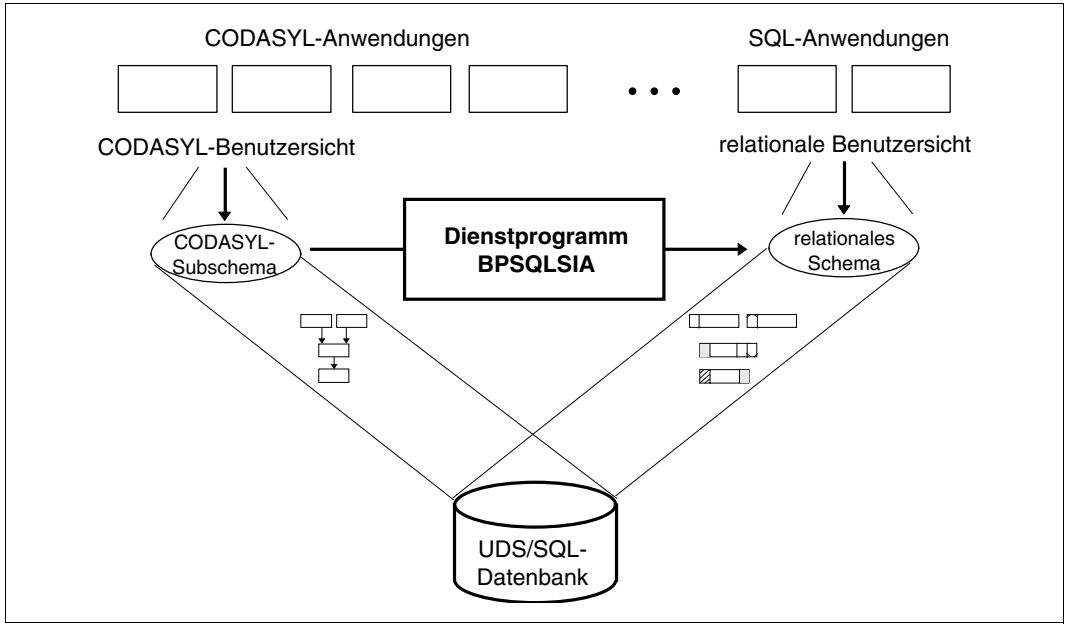


Bild 4: Zwei Benutzersichten derselben UDS/SQL-Datenbank

Zusammenfassend bietet UDS/SQL folgende Möglichkeiten, Programmschnittstellen und Datenorganisationen zu kombinieren:

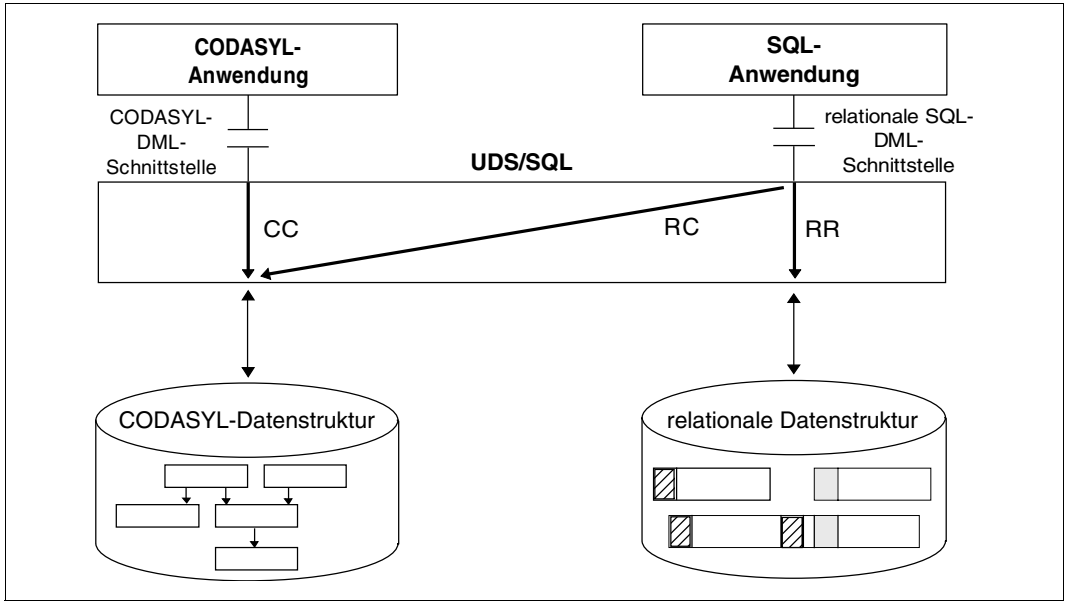


Bild 5: Koexistenz von Schnittstellen und Datenmodellen in UDS/SQL

Relationale SQL-Programmschnittstelle → CODASYL-Datenorganisation (RC)

Über die relationale SQL-Programmschnittstelle greifen Anwendungen auf eine CODASYL-Datenorganisation zu, für die eine relationale Sicht in der oben beschriebenen Weise erzeugt wurde.

Der Anwendungsfall RC gestattet es, die implizit vorhandenen Set-Beziehungen auszunutzen. Der SQL-Programmierer braucht die CODASYL-Strukturen jedoch nicht zu kennen. Durch das Ausnutzen der Set-Beziehungen gewährleistet UDS/SQL die Integrität der Datenbeziehungen (referentielle Integrität).

Relationale SQL-Programmschnittstelle → relationale Datenorganisation (RR)

Über die relationale SQL-Programmschnittstelle greifen Anwendungen auf eine rein relationale Datenorganisation zu.

Der Anwendungsfall RR bietet sich für kompatible Programme an, da hier die Primär- und Fremdschlüssel normgemäß vom Anwender verwaltet werden und damit der Handhabung bei SESAM/SQL und INFORMIX entsprechen. Da sich in diesem Fall die Datenstrukturen nicht auf Set-Beziehungen abstützen, kann die referentielle Integrität nicht vom System gewährleistet werden.

CODASYL-Programmschnittstelle → CODASYL-Datenorganisation (CC)

Über die CODASYL-Programmschnittstelle (COBOL-DML oder CALL-DML von UDS/SQL) greifen Anwendungen auf die CODASYL-Datenstrukturen zu.

Koexistenz von Schnittstellen und Datenmodellen

UDS/SQL bietet die beschriebenen Möglichkeiten **koexistent**, d.h. ein Database Handler kann gleichzeitig folgende Aufgaben durchführen:

– SQL-Anwendungen und CODASYL-Anwendungen bedienen (SQL- und CODASYL-Sprachelemente können sogar in ein und derselben Anwendung gemischt werden, nur nicht innerhalb einer Transaktion)

– relationale Datenstrukturen und CODASYL-Datenstrukturen bearbeiten

– alle drei oben beschriebenen Anwendungsfälle unterstützen (RR, RC und CC)

Im Datenmodell von UDS/SQL sind also das relationale und das CODASYL-Datenmodell in der Weise realisierbar, dass jedes Modell für sich unabhängig existieren kann und dass zusätzlich SQL-Anweisungen auf CODASYL-Datenstrukturen abgebildet werden können. Diese Eigenschaft bezeichnet man als **Koexistenzmodell**, eine Eigenschaft, die in UDS/SQL als Erstes realisiert wurde und die Anwendungsmöglichkeiten erlaubt, welche die Flexibilität und Einfachheit des relationalen Datenmodells mit der Effizienz und Performanz von Netzwerk-Datenbanksystemen vereinen.

Einsatzbereiche der Schnittstellen

Typische Einsatzbereiche für die COBOL-DML bzw. CALL-DML-Schnittstelle sind:

– High-End-OLTP-Anwendungen und extrem performanzkritische Anwendungen (Online Transaction Processing)

– spezielle Anwendungen, die mit Netzwerkstrukturen besonders effizient ablaufen, z.B. Stücklistenverarbeitung

Die SQL-Schnittstelle wird in folgenden Anwendungsfällen eingesetzt:

– Objektorientierte Datenbanksysteme
– Client-/Server-Anwendungen
– Data Warehouse
– Anwendungen, die portabel und standardkonform erstellt werden sollen
– 4GL-Programmierung mit DRIVE (4th Generation Language)
– COBOL-Programme, die Datenbankzugriffe enthalten sollen

2.3 Das Universelle Datenbank-System UDS/SQL

Das **U**niverselle **D**atenbank-**S**ystem UDS/SQL bietet dem Benutzer vielfältige Strukturierungsmöglichkeiten und eine Fülle von Leistungen für Datenbankaufbau, Produktiveinsatz und Datenbankpflege:

- Strukturierungsmöglichkeit

  Die Datenbankstruktur ist das Abbild der organisatorischen, betriebswirtschaftlichen und technischen Zusammenhänge eines Unternehmens. Mit UDS/SQL kann man relationale, hierarchische und vernetzte Strukturen abbilden. Eine Anpassung des Datenbankschemas an die gegebenen Strukturen ist also problemlos möglich.

  Die Datenstruktur lässt sich neuen Bedingungen anpassen, wenn z.B. Zusammenhänge geändert werden oder die Aufgabenstellung erweitert wird.

- Redundanzfreies Speichern der Daten

  Daten werden bei UDS/SQL redundanzfrei gespeichert. Somit wird der benötigte Speicherplatz minimiert und der Änderungsdienst vereinfacht.
  Bei einer Änderung muss der Datenbestand nur an einer Stelle aktualisiert werden. Die Daten stehen damit immer allen Anwendungen im aktuellen Zustand und in der gleichen Form zur Verfügung.

- Datenunabhängigkeit der Programme

  Bei UDS/SQL werden auf logischer Ebene die Datensätze für die gesamte Datenbank nur einmal beschrieben. Die Datenstruktur wird beim Aufbau der Datenbank festgelegt und ist verbindlich für alle Anwendungsprogramme.
  Diese Anwendungsprogramme operieren auf der logischen Ebene. Änderungen in der physischen Organisation berühren die Programme nicht. Physische Gesichtspunkte (etwa beim Speichern) sind also von der Fachabteilung entkoppelt. Der Anwender kennt nur die logische Struktur der Daten, die für seine Arbeit notwendig sind.
  Die Programmerstellung und die Wartung werden vereinfacht und vereinheitlicht.

- Flexible Auswertung der Daten

  Der Benutzer kann die Daten unter verschiedenen Aspekten auswerten, da UDS/SQL die Datenbestände zentral führt. Datensätze und Satzelemente können nicht nur über definierte Schlüssel ausgewählt werden, sondern auch über beliebig komplexe Bedingungen und beliebige Satzelementinhalte.

- Zentrale Datenschutzmaßnahmen

  UDS/SQL garantiert durch wirksame, flexibel einsetzbare Schutzmechanismen, dass für jede Benutzergruppe nur genau definierte Teile und Ausschnitte aus der Datenbank zugänglich sind.
  UDS/SQL prüft die Zugriffsberechtigung des Benutzers, bevor Anforderungen an die Datenbank tatsächlich ausgeführt werden.

- Gleichzeitiger Zugriff auf die Daten durch viele Anwender

  Bei UDS/SQL können viele Benutzer gleichzeitig die anwendungsneutral gespeicherten Daten nutzen. Ihre Anforderungen werden „zeitlich verzahnt“ abgearbeitet. Somit wird der Gesamtdurchsatz optimiert.
  UDS/SQL enthält außerdem Funktionen, die eine gegenseitige Behinderung von Benutzern abfangen, wenn sie konkurrierend auf denselben Datensatz zugreifen.

- Zentrale Sicherungsmaßnahmen

  Das Datensicherungskonzept von UDS/SQL enthält Vorkehrungen, um die Datenbestände vor Zerstörung und Verlust zu schützen. Zerstörte Datenbestände können somit wieder in den letzten konsistenten Zustand gebracht werden.

- Unterstützung von Spiegelplatten

  UDS/SQL unterstützt DRV (Dual Recording by Volume). Bei Einsatz von DRV werden die Daten vom Betriebssystem gleichzeitig auf zwei Platten geschrieben. Damit ist eine erhöhte Verfügbarkeit und Ausfallsicherheit gewährleistet. Denn auch bei Ausfall einer Platte stehen den Anwendungen weiterhin alle Daten konsistent und im aktuellen Zustand zur Verfügung.

- Unterstützung der BS2000-Zugriffsmethode FASTPAM

  UDS/SQL unterstützt die BS2000-Zugriffsmethode FASTPAM. FASTPAM gewährleistet einen besonders performanten Zugriff auf Dateien und Realms der Datenbank.

- Unterstützung der unterbrechungsfreien Zeitumstellung

  Die Unterstützung der unterbrechungsfreien Zeitumstellung gewährleistet den störungsfreien Ablauf von UDS/SQL bei der saisonbedingten Umstellung der Lokalzeit, z.B. Umstellung von Mitteleuropäischer Sommerzeit (MESZ) auf Mitteleuropäische Zeit (MEZ).

3 Entwurf der Datenbank

Voraussetzung für die Definition einer Datenbank mit produktspezifischen Sprachmitteln ist eine genaue und eingehende Analyse der Daten, der Beziehungen zwischen den Daten, der Abhängigkeiten untereinander und der Benutzeranforderungen. Diese Untersuchungen sollten möglichst gründlich durchgeführt werden, da sie von entscheidender Bedeutung sind für die nachfolgenden Arbeiten. Bekanntlich sind Design-Fehler die folgenschwersten, da sie später nur sehr aufwändig behoben werden können.

3.1 Datenmodellierung

Zur Datenmodellierung existieren vielfältige Modelle und Hilfsmittel. Zu den wichtigsten zählen das Entity-Relationship-Modell (ERM) und die Structured Analysis (SA). Zum Beispiel kann ein Data Dictionary eingesetzt werden, um gesammelte Daten zu erfassen und zu verwalten. Weiterführende Literatur und Schulung informieren wesentlich ausführlicher über Datenanalyse und -Design, als es in einem produktspezifischen Handbuch möglich und sinnvoll ist.

Ganz grob umfasst die Analyse folgende Schritte:

– Abgrenzung der Miniwelt von der realen Welt

  Es wird festgelegt, welcher Ausschnitt aus der realen Welt Grundlage sein soll für das entstehende Datenmodell, um die Anforderungen in DV-Verfahren umsetzen zu können.

– Informationsanalyse

  Aufgabe der Informationsanalyse ist es, die Daten und Informationen der Miniwelt sowie die wechselseitigen Beziehungen der Daten untereinander zu untersuchen.

– Funktionsanalyse

  Sie dokumentiert, welche Daten die einzelnen Anwendungen in welcher Reihenfolge benötigen.

Am Ende des Analyseprozesses steht ein Modell, das den untersuchten Ausschnitt der realen Welt so beschreibt, dass die Daten mit einem Datenbanksystem verwaltet werden können. Die Daten sind vollständig, widerspruchsfrei und liegen in normalisierter Form vor. Die Art der Beziehungen zwischen den Daten ist definiert, z.B. eine 1:n-Beziehung zwischen Kunde und Auftrag.

Die logische Struktur der Daten wird auch konzeptionelles Schema genannt. Das konzeptionelle Schema ist eine wichtige Arbeitsgrundlage sowohl für die Datenbank-Designer als auch für die Fachabteilungen. Das konzeptionelle Schema stellt die Daten und ihre Beziehungen dar, ohne festzulegen, in welchem Datenmodell die beschriebene Struktur abgebildet wird.

Wenn ein konzeptionelles Schema existiert, muss untersucht werden, wie die Verteilung der Daten auf Datenbanken und auf Rechner aussehen soll.

3.2 Verteilung der Daten

Der Datenbestand kann auf mehrere Datenbanken aufgeteilt werden, z.B. eine Personaldatenbank für die Lohn- und Gehaltsabrechnung, eine Kundendatenbank für die Auftragsabwicklung usw. Die Datenbanken können von einem oder mehreren DBHs bearbeitet werden. Datenbanken können auch auf verschiedene Rechner verteilt sein.

Bei der Aufteilung des Datenbestandes können folgende Gesichtspunkte eine Rolle spielen:

– Zusammenhänge zwischen den Anwendungen
– organisatorische Gegebenheiten
– Anforderungen an Verfügbarkeit und Datenmengen

Ein DBH - mehrere Datenbanken

Diese Konstellation wird auch Multi-DB-Betrieb genannt. Viele Anwenderprogramme arbeiten mit mehreren Datenbanken. Ein Anwenderprogramm (AP) kann innerhalb einer Transaktion auf mehrere Datenbanken zugreifen. Der DBH steuert die Bearbeitung der Datenbanken und gewährleistet die Konsistenz des gesamten Datenbestandes.

Die Datenbanken, die von *einem* DBH bearbeitet werden, gehören zu einer Datenbank-Konfiguration.

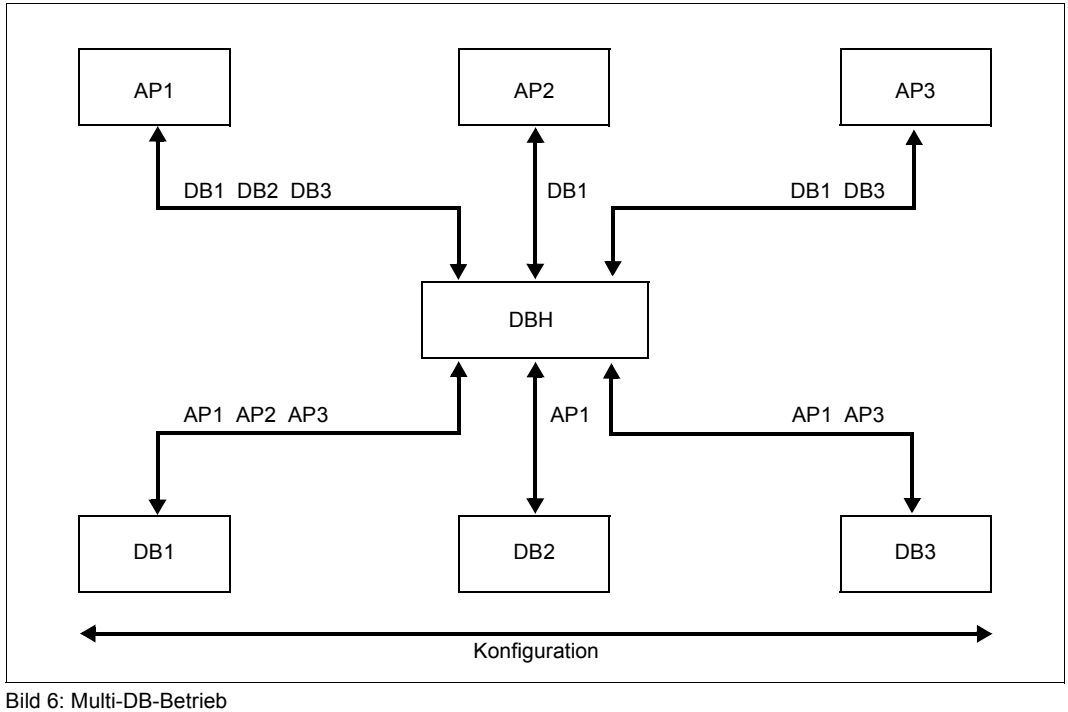


Bild 6: Multi-DB-Betrieb

Gründe für die Aufteilung auf mehrere Datenbanken:

– Daten, die nur zu bestimmten Zeiten oder für bestimmte Verfahren gebraucht werden, können in einer eigenen Datenbank liegen; diese Datenbank muss dann nicht immer angeschlossen sein.

– Eine umfangreiche Datenbankanwendung lässt sich leichter verwirklichen durch schrittweises Entwickeln zusätzlicher Datenbanken.

Mehrere DBHs - mehrere Datenbanken

Ein Multi-DB-Betrieb ist auch mit Datenbanken möglich, die anderen Konfigurationen angehören. Andere Konfigurationen können auf einem anderen Verarbeitungsrechner eines homogenen BS2000-Rechnernetzes liegen (siehe Bild 7) oder auf dem gleichen Rechner (siehe Bild 8).
Beim Einsatz mehrerer DBHs ist das UDS/SQL-Zusatzprodukt UDS-D erforderlich (siehe Handbuch „Datenbankbetrieb“). Mit UDS-D kann ein COBOL- oder CALL-DML-Anwenderprogramm, nicht jedoch ein SQL-Programm, innerhalb einer Transaktion auf Datenbanken mehrerer Konfigurationen zugreifen. Dabei benötigt es keinerlei Informationen über den Ort der angesprochenen Datenbanken. Eine Erweiterung des Multi-DB-Betriebs auf den UDS-D-Betrieb ist somit ohne Anpassungen bestehender Anwenderprogramme möglich. Die konfigurationsübergreifende Konsistenz der Datenbanken ist zu jedem Zeitpunkt der Verarbeitung gesichert. Ebenso sorgt UDS-D für eine konfigurationsübergreifende Deadlock-Überwachung und Deadlock-Auflösung.

Die folgenden Bilder zeigen die zusätzlichen Möglichkeiten mit UDS-D. Sie sind als Ergänzung zu Bild 6 zu verstehen.

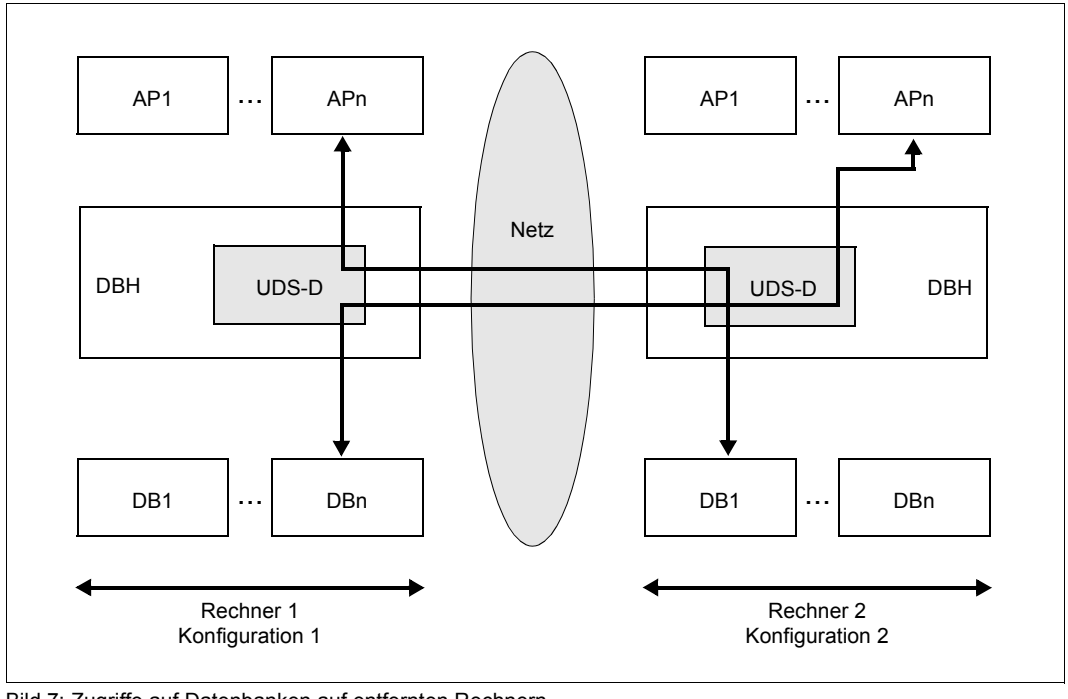


Bild 7: Zugriffe auf Datenbanken auf entfernten Rechnern

Gründe, Datenbanken auf mehrere Rechner in einem Netz zu verteilen:

– Anpassung

  Die Arbeitsabläufe können sich am Rechenzentrum vor Ort orientieren und die Datenhaltung kann noch besser an die Organisation des Unternehmens angepasst werden.

– Verfügbarkeit

  Bei verteilten Datenbanken ist der Datenbestand leichter verfügbar, da besonders wichtige Daten gekoppelt in mehr als einer Anlage gespeichert werden können. Werden die Sicherungen an verschiedenen Orten gespeichert, ist bei einem Rechnerausfall immer noch der Zugriff auf die Datenbestände in anderen Rechnern möglich.

– Kostenreduktion

  Es werden Kosten eingespart, wenn weniger Datenstationen eine ständige Verbindung zum zentralen Rechenzentrum brauchen. Die meisten Zugriffe sind lokal möglich; nur gelegentlich wird auf entfernte Datenbestände zugegriffen.

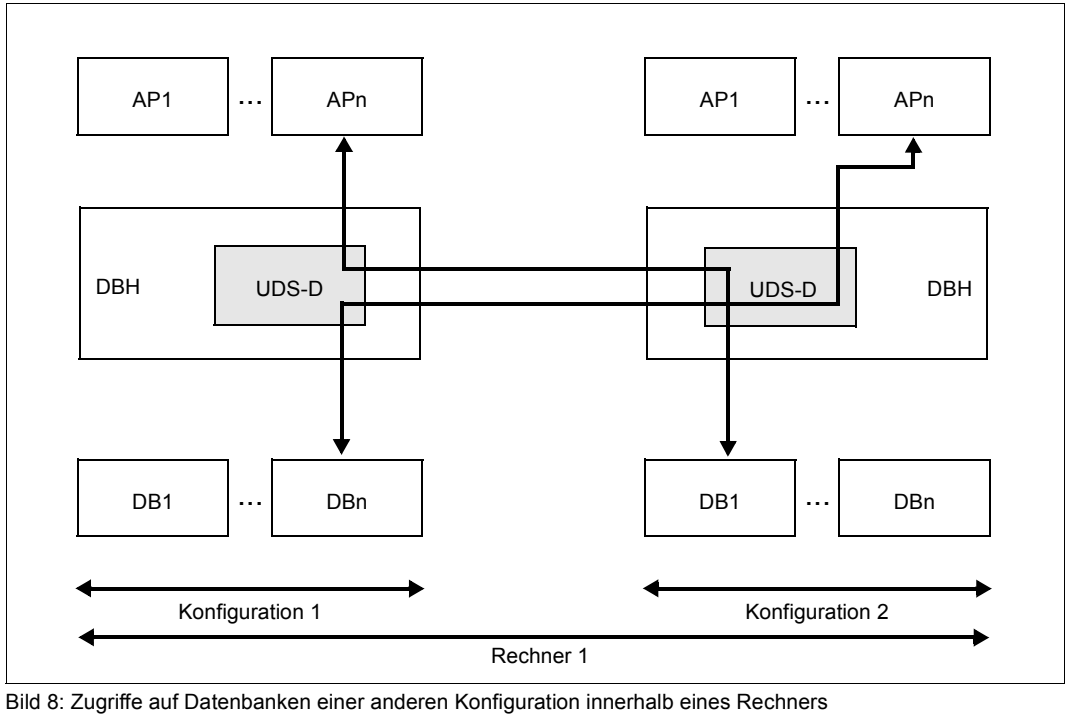


Bild 8: Zugriffe auf Datenbanken einer anderen Konfiguration innerhalb eines Rechners

Gründe, mehrere Konfigurationen auf einem Rechner zu betreiben:

- Performanzverbesserung
- höhere Unabhängigkeit zwischen Anwendungen
- getrennte Administration
- getrennte Abrechnung
- getrennte Verantwortlichkeiten

3.3 Technische Umsetzung

3.3.1 Logische Struktur einer UDS/SQL-Datenbank definieren

Die logische Struktur einer UDS/SQL-Datenbank, das UDS/SQL-Schema, kann entweder auf Basis des CODASYL-Konzepts oder auf Basis des relationalen Konzepts definiert werden.

CODASYL-Datenbank-Entwurf

Bei einer CODASYL-Datenbank definieren Sie die Datenbeziehungen über die Strukturen der Datenbank. Ein wesentlicher Vorteil dieses Konzeptes liegt darin, dass zum Ablaufzeitpunkt diese Datenbeziehungen vom Database Handler automatisch auf ihre Konsistenz überprüft werden.

Das Sprachmittel zur Definition einer Datenbank ist die Schema-DDL (Data Description Language).

Der Aufbau einer entsprechenden Datenbank über die Schema-DDL beinhaltet folgende Informationsobjekte:

- Datenbankbereiche (Realms)
- Satzarten (Records)
- Set-Beziehungen
- Schlüssel als optimierte Zugriffspfade
- Sortierungen
- Set-Mitgliedschaft
- Set-Auswahl

Die genaue Beschreibung der Schema-DDL finden Sie ab Seite 49.

Relationaler Datenbank-Entwurf

Bei einer relationalen Datenbank werden die Datenbeziehungen über die Verknüpfung der Werte zum Ablaufzeitpunkt hergestellt.

Sie definieren mit der Schema-DDL die Bestandteile einer relationalen Struktur, nämlich Tabellen (Satzarten) und Felder.

Um die Beziehungen zwischen den Tabellen zum Ausdruck zu bringen, ist es notwendig, die Primärschlüssel der jeweiligen Tabellen als Fremdschlüssel in den korrespondierenden Tabellen zusätzlich zu definieren.
Diese Primär- und Fremdschlüssel müssen Sie selbst vergeben und auf Eindeutigkeit der Beziehungen überprüfen.
Diese Beziehungen müssen Sie selbst zum Ablaufzeitpunkt im Programm überprüfen.

Eine detaillierte Beschreibung der Methoden, ein konzeptionelles Schema in relationale Strukturen umzusetzen, finden Sie in der Datenbankliteratur oder können Sie in entsprechenden Kursen lernen.

3.3.2 Physische Struktur einer UDS/SQL-Datenbank festlegen

Mit der SSL (Storage Structure Language) legen Sie die physische Datenstruktur fest.

Haupsächlich beinhaltet die SSL:

– Mengenangaben
– Datenplatzierung
– Optimierungsfestlegungen
– Verknüpfungsfestlegungen: interne Tabellen, Ketten, Listen.

Die Daten werden im Wesentlichen über die SSL auf Datenbankbereiche aufgeteilt, und zwar nach folgenden Gesichtspunkten:

– in Abhängigkeit von Zugriffshäufigkeiten
– nach physischen Speichermöglichkeiten (Plattenkapazität)
– nach angewendeten Sicherungsverfahren

3.3.3 Benutzersichten

Subschema

Mit der Subschema-DDL wählen Sie die Teile aus der Datenbank aus, die gezielt für einen Anwendungsfall notwendig sind.

Die genaue Beschreibung der Subschema-DDL finden Sie ab Seite 181.

Relationales Schema

In einer SQL-Anwendung arbeiten Sie mit einem relationalen Schema. Ein relationales Schema ist die relationale Sicht auf ein existierendes Subschema. Eine solche Sicht können Sie mit dem Dienstprogramm BPSQLSIA aus einem Subschema ableiten.

Weitere Informationen zum relationalen Schema finden Sie auf Seite 194.

4 Schema-DDL

4.1 Einführung

Voraussetzung für den Entwurf eines UDS/SQL-Schemas ist eine sorgfältige Analyse der geplanten Datenbankanwendungen und der dafür benötigten Informationen. Die Analyse muss nicht nur die benötigten Informationen vollständig liefern, sondern auch die Beziehungen der Informationen untereinander.

Die so gewonnene Datenstruktur übersetzen Sie mit der Schema-DDL in ein UDS/SQL-Schema.

Im UDS/SQL-Schema müssen alle Daten definiert sein, die für die Aufgaben, die mit Hilfe der Datenbank gelöst werden sollen, notwendig sind. Das UDS/SQL-Schema hat jedoch keine direkte Schnittstelle zum Benutzer; Aspekte der benutzerfreundlichen Aufbereitung der Daten brauchen beim Entwurf des UDS/SQL-Schemas nicht berücksichtigt zu werden.

Das CODASYL-Modell sieht insgesamt folgende Dateneinheiten vor, die Sie zur logischen Datenstrukturierung definieren können:

Feld Kleinste benennbare Dateneinheit innerhalb einer Satzart. Es ist definiert durch Feldtyp, Feldlänge und Feldname.

Vektor Feld mit Wiederholungsfaktor. Der Wiederholungsfaktor muss größer als 1 sein. Er gibt an, wie viel Duplikate des Feldes zu dem Vektor zusammengefasst werden.

Datengruppe
Benennbare Zusammenfassung von Satzelementen innerhalb einer Satzart. Ein Satzelement kann dabei ein Feld, ein Vektor oder selbst eine Datengruppe sein. Die Definition von Datengruppen, die keine Wiederholungsgruppen sind, ist nur für Subschemata erlaubt.

Wiederholungsgruppe
Datengruppe mit Wiederholungsfaktor. Der Wiederholungsfaktor muss größer als 1 sein. Er gibt an, wie viel Duplikate der Datengruppe zu der Wiederholungsgruppe zusammengefasst werden.

Satzart
: Benennbare Zusammenfassung von Satzelementen und kleinste Dateneinheit, die UDS/SQL über einen eindeutigen Identifizierer verwaltet.
Satzelement kann dabei ein Feld, ein Vektor oder eine Wiederholungsgruppe sein.

Set
: Benennbare Beziehung zwischen zwei Satzarten.

Realm
: Benennbare physische Untereinheit der Datenbank. Dient als Verwaltungseinheit für Datenschutz, Datensicherung und Steuerung des konkurrierenden Zugriffs.

Die Sprachelemente der DDL, mit denen Sie die Dateneinheiten definieren, sind vorgestellt auf den Seiten 51 bis 65.

Die verwendete Metasprache ist auf Seite 18 erklärt, die allgemeinen Syntaxregeln sind auf Seite 230 zusammengefasst.

Die Schema-DDL übersetzen Sie mit dem DDL-Compiler. Zur Bedienung des DDL-Compilers siehe Handbuch „Aufbauen und Umstrukturieren“, Schema-DDL übersetzen).

4.2 Definieren eines Feldes

Ein Feld ist die kleinste Dateneinheit innerhalb einer Satzart, für die Sie mit der Schema-DDL einen Namen vergeben können. Der Begriff Feld steht gleichzeitig für die Gesamtheit aller Werte, die das Feld aufnehmen kann. Ein bestimmter Wert, der in einem Feld gespeichert ist, heißt dessen Feldinhalt, die Gesamtheit aller möglichen Feldinhalte heißt Wertebereich des Feldes.

Ein Feld definieren bedeutet deshalb hauptsächlich, den Wertebereich des Feldes zu definieren. Außerdem bestimmen Sie in der Feldbeschreibung auch, in welcher physischen Form ein Feldinhalt gespeichert wird.

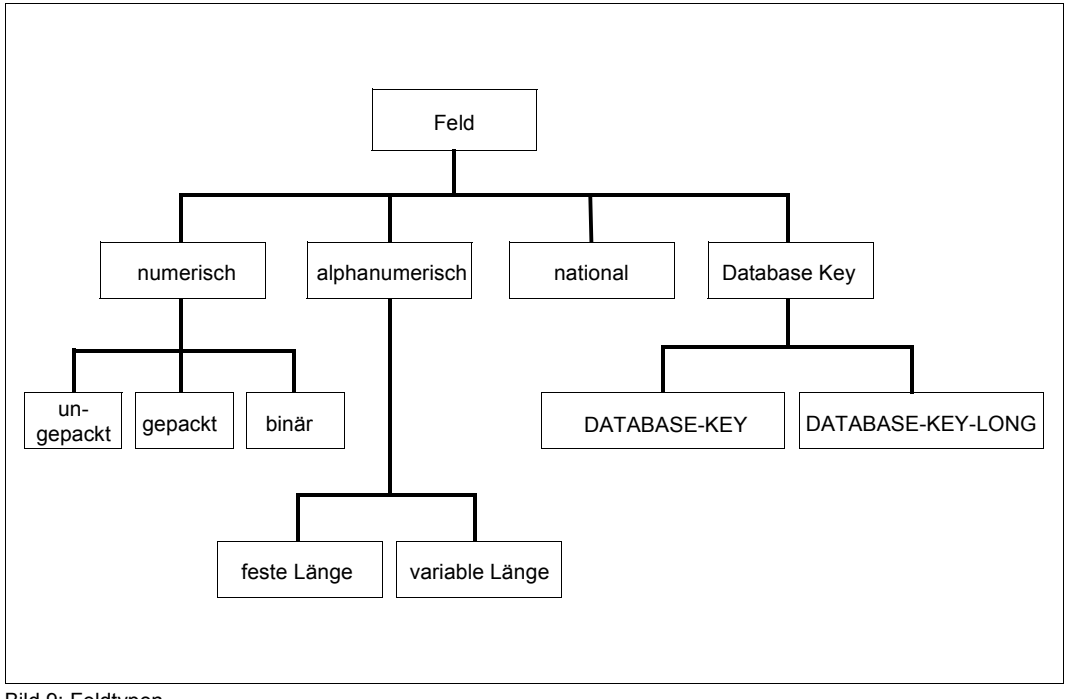


Bild 9: Feldtypen

Im Bild 9 sind die verschiedenen Möglichkeiten zur physischen Speicherung von Feldinhalten dargestellt.

4.2.1 Definieren eines ungepackten numerischen Feldes

```
[stufennummer ]feldname PICTURE IS maskenzeichenkette.
```

Ungepackte Felder nehmen numerische Werte auf. Mit ungepackten Feldern kann der Datenbankprogrammierer rechnen, und er kann die Felder ausdrucken lassen.

Über *stufennummer* bestimmen Sie, ob das Feld einer Wiederholungsgruppe angehört:

Soll das Feld nicht zu einer Wiederholungsgruppe gehören, müssen Sie die Stufennummer so wählen, dass in der gesamten Satzart keine kleinere Stufennummer vorkommt. Dies gilt insbesondere, wenn Sie das Feld als Schlüsselfeld benutzen wollen.
Soll das Feld Bestandteil einer Wiederholungsgruppe sein, gehen Sie gemäß Seite 62 vor.

Standardwert für *stufennummer* ist 1.

Mit *feldname* geben Sie dem Feld einen Namen.

Mit *maskenzeichenkette* definieren Sie den Wertebereich des Feldes in Form einer Maske, d.h. durch eine symbolische Darstellung der möglichen Feldinhalte.
maskenzeichenkette darf aus folgenden Symbolen bestehen:

Symbol	Bezeichnung	Erläuterung
S	Vorzeichensymbol	müssen Sie angeben, wenn Sie positive und negative Feldinhalte unterscheiden wollen. Wenn Sie kein S angeben, fasst UDS/SQL den Feldinhalt immer als positive Zahl auf. Dieses Symbol geben Sie höchstens einmal und dann an der ersten Stelle der Maskenzeichenkette an.
9	Ziffernsymbol	ist das einzige Symbol der Maskenzeichenkette, das eine physische Speicherstelle symbolisiert. Diese Speicherstelle kann eine Dezimalziffer aufnehmen. Sie können das Ziffernsymbol mehrmals wiederholen. Die Anzahl der Wiederholungen bestimmt somit die Größe des Feldes.

Tabelle 4: Symbole der *maskenzeichenkette* beim Definieren eines ungepackten numerischen Feldes (Teil 1 von 2)

Symbol	Bezeichnung	Erläuterung
V	Dezimalpunktsymbol	markiert an der durch Ziffernsymbole 9 definierten Maske die Stelle des Dezimalpunktes. Wenn Sie kein V angeben, wird standardmäßig ein V an der letzten Stelle der Maske angenommen.
P		müssen Sie verwenden, wenn der Dezimalpunkt mehr als eine Stelle außerhalb der durch Ziffernsymbole 9 definierten Maske liegt und Sie ihn deshalb nicht mit „V“ kennzeichnen können. P steht für eine Null, die UDS/SQL zwischen der durch Ziffernsymbole 9 definierten Maske und dem Dezimalpunkt einfügen soll. P können Sie mehrmals wiederholen.
(*ganzzahl*)	Wiederholungssymbol	Das Ziffernsymbol 9 und das Dezimalpunktsymbol P dürfen Sie wiederholen. Dazu können Sie die Zeichen entweder mehrmals hintereinander angeben oder Sie geben das Zeichen nur einmal an, gefolgt von einem Wiederholungssymbol: Mit *ganzzahl* geben Sie den Wiederholungsfaktor an.

Tabelle 4: Symbole der *maskenzeichenkette* beim Definieren eines ungepackten numerischen Feldes (Teil 2 von 2)

Maximal können Sie das Feld für die Aufnahme von 18 Ziffern definieren.

Wenn für ein ungepacktes numerisches Feld mindestens eine der folgenden Aussagen zutrifft, können Sie nicht mit SQL auf das Feld zugreifen und auch keinen neuen Satz der betroffenen Satzart einfügen:

– Anzahl Speicherstellen > 15
– Skalenfaktor < 0
– Skalenfaktor > Anzahl Speicherstellen

Ein positiver Skalenfaktor gibt die Anzahl der Stellen rechts vom Dezimalpunkt an, ein negativer Skalenfaktor gibt an, wie viele Nullen UDS/SQL beim Rechnen an den Feldinhalt anfügen muss.

4.2.2 Definieren eines gepackten numerischen Feldes

```
[stufennummer ]feldname TYPE IS FIXED REAL DECIMAL
                        [ ganzzahl-1[,ganzzahl-2]].
```

Gepackte Felder nehmen numerische Werte auf. Sie dienen dem Datenbankprogrammierer ausschließlich als Rechenfelder und lassen sich nicht ohne zusätzliche Aufbereitung durch ein DML-Programm ausdrucken. Gepackte Felder benötigen weniger Speicherplatz und können schneller verarbeitet werden als ungepackte Felder.

Über *stufennummer* bestimmen Sie, ob das Feld einer Wiederholungsgruppe angehört: Soll das Feld nicht zu einer Wiederholungsgruppe gehören, müssen Sie die Stufennummer so wählen, dass in der gesamten Satzart keine kleinere Stufennummer vorkommt. Dies gilt insbesondere, wenn Sie das Feld als Schlüsselfeld benutzen wollen.

Soll das Feld Bestandteil einer Wiederholungsgruppe sein, gehen Sie gemäß Seite 62 vor.

Mit *feldname* geben Sie dem Feld einen Namen.

ganzzahl-1 und *ganzzahl-2* beschreiben den Wertebereich des Feldes:

– *ganzzahl-1* gibt die Anzahl der Speicherstellen des Feldes an, wobei maximal 18 möglich sind. Jede Speicherstelle kann eine Dezimalziffer aufnehmen.
  Standardwert für *ganzzahl-1* ist 18.

– *ganzzahl-2* gibt die Stellung des Dezimalpunktes an, was aber keinen Einfluss auf die Anzahl der Speicherstellen hat. Ist *ganzzahl-2* positiv, dann gibt sie die Stellenanzahl rechts vom Dezimalpunkt an; ist sie negativ, gibt sie an, wie viele Nullen UDS/SQL beim Rechnen an den Feldinhalt anfügen muss.
  Wird *ganzzahl-2* mit m bezeichnet, so bedeutet die Angabe von *ganzzahl-2* also eine Multiplikation des Feldinhaltes mit 10^{-m}.
  Standardwert für *ganzzahl-2* ist 0.

Wenn für ein gepacktes numerisches Feld mindestens eine der folgenden Aussagen zutrifft, können Sie nicht mit SQL auf das Feld zugreifen und auch keinen neuen Satz der betroffenen Satzart einfügen:

– Anzahl Speicherstellen > 15
– Skalenfaktor < 0
– Skalenfaktor > Anzahl Speicherstellen

Ein positiver Skalenfaktor gibt die Anzahl der Stellen rechts vom Dezimalpunkt an, ein negativer Skalenfaktor gibt an, wie viele Nullen UDS/SQL beim Rechnen an den Feldinhalt anfügen muss.

4.2.3 Definieren eines binären Feldes

```
[stufennummer ]feldname TYPE IS FIXED REAL BINARY[ {15 | 31} ].
```

Binäre Felder nehmen ganzzahlige numerische Werte auf. Sie dienen dem Datenbankprogrammierer ausschließlich als Rechenfelder und lassen sich nicht ohne zusätzliche Aufbereitung durch ein DML-Programm ausdrucken. Binäre Felder benötigen weniger Speicherplatz und können schneller verarbeitet werden als gepackte und ungepackte Felder.

Wenn Sie die Ausrichtung von BINARY 15-Feldern auf Halbwortgrenze und BINARY 31-Feldern auf Wortgrenze nicht vornehmen, richtet UDS/SQL die Felder automatisch aus.

Über *stufennummer* bestimmen Sie, ob das Feld einer Wiederholungsgruppe angehört:
Soll das Feld nicht zu einer Wiederholungsgruppe gehören, müssen Sie die Stufennummer so wählen, dass in der gesamten Satzart keine kleinere Stufennummer vorkommt.
Dies gilt insbesondere, wenn Sie das Feld als Schlüsselfeld benutzen wollen.

Soll das Feld Bestandteil einer Wiederholungsgruppe sein, gehen Sie gemäß Seite 62 vor.

Mit *feldname* geben Sie dem Feld einen Namen.

Mit BINARY 15 definieren Sie ein binäres Feld mit dem Wertebereich -2^{15} bis $2^{15}-1$.

Mit BINARY 31 definieren Sie ein binäres Feld mit dem Wertebereich -2^{31} bis $2^{31}-1$.

Wenn Sie keine Zahl angeben, wird standardmäßig BINARY 15 angenommen.

4.2.4 Definieren eines alphanumerischen Feldes fester Länge

```
[stufennummer ]feldname { PICTURE IS maskenzeichenkette }
                        { TYPE IS CHARACTER ganzzahl    } .
```

Alphanumerische Felder können beliebige Zeichen aufnehmen.

Über *stufennummer* bestimmen Sie, ob das Feld einer Wiederholungsgruppe angehört: Soll das Feld nicht zu einer Wiederholungsgruppe gehören, müssen Sie die Stufennummer so wählen, dass in der gesamten Satzart keine kleinere Stufennummer vorkommt. Dies gilt insbesondere, wenn Sie das Feld als Schlüsselfeld benutzen wollen. Soll das Feld Bestandteil einer Wiederholungsgruppe sein, gehen Sie gemäß Seite 62 vor.

Mit *feldname* geben Sie dem Feld einen Namen.

maskenzeichenkette darf aus folgenden Symbolen bestehen:

Symbol	Bezeichnung	Erläuterung
X		symbolisiert eine Speicherstelle, die jedes beliebige Zeichen des Zeichenvorrats aufnimmt.
A		symbolisiert eine Speicherstelle, die nur einen Buchstaben oder einen Zwischenraum aufnehmen soll. UDS/SQL unterscheidet jedoch nicht zwischen A und X.
9	Ziffernsymbol	symbolisiert eine Speicherstelle, die nur eine Ziffer aufnimmt. 9 darf nicht links von einem A oder X stehen.
(*ganzzahl*)	Wiederholungssymbol	Jedes der Symbole X, A oder 9 können Sie wiederholen, indem Sie es mehrmals hintereinanderschreiben oder es nur einmal schreiben, gefolgt von dem Wiederholungssymbol: *ganzzahl* bezeichnet dabei den Wiederholungsfaktor. Standardwert ist 1; Maximalwert ist 18 bei 9 und 255 bei X und A.

Tabelle 5: Symbole der *maskenzeichenkette* beim Definieren eines alphanumerischen Feldes fester Länge

Das erste Symbol in der Maskenzeichenkette muss A oder X sein. (Eine mit 9 beginnende Maskenzeichenkette definiert ein numerisches Feld).

Maximal können Sie das Feld für die Aufnahme von 255 Zeichen definieren. Dabei dürfen Sie das Ziffernsymbol höchstens 18 mal wiederholen und die Maskenzeichenkette darf nur aus höchstens 30 Zeichen bestehen.

Mit *ganzzahl* geben Sie die Anzahl der Speicherstellen des Feldes an, wobei jede Speicherstelle ein Zeichen des Zeichenvorrats aufnehmen kann.

4.2.5 Definieren eines alphanumerischen Feldes variabler Länge

```
[stufennummer ]feldname-1 {PICTURE IS LX(ganzzahl)   }
                          {TYPE IS CHARACTER ganzzahl}
                              DEPENDING ON feldname-2.
```

Alphanumerische Felder variabler Länge können beliebige Zeichen aufnehmen.

Über *stufennummer* bestimmen Sie, ob das Feld einer Wiederholungsgruppe angehört. Da variable Felder nicht Bestandteil einer Wiederholungsgruppe sein dürfen, müssen Sie die Stufennummer so wählen, dass in der gesamten Satzart keine kleinere Stufennummer existiert.

Mit *feldname-1* geben Sie dem Feld einen Namen.

Mit *ganzzahl* bestimmen Sie die maximale Länge des variablen Feldes. *ganzzahl* steht also für die maximale Anzahl von Speicherstellen, wobei jede Speicherstelle ein beliebiges Zeichen des Zeichenvorrats aufnehmen kann. Für *ganzzahl* müssen Sie einen Wert > 0 angeben.

Mit *feldname-2* nennen Sie den Namen eines Feldes, das Sie unmittelbar vor dem variablen Feld als binäres Feld mit dem Wertebereich $0\text{-}2^{15}\text{-}1$ definiert haben müssen. Vor dem Einspeichern oder Ändern eines Satzes mit variablem Feld muss der Datenbankprogrammierer im Feld *feldname-2* die aktuelle Länge des variablen Feldes angeben. Diese (aktuelle) Länge kann auch 0 sein. In diesem Fall wird in der Datenbank weder das variable Feld noch das Satzlängenfeld abgespeichert.

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, darf die maximale Länge eines Satzes mit variablem Feld folgenden Wert nicht überschreiten:

– 2012 byte bei 2048 byte Seitenlänge (2-Kbyte-Seitenformat)
– 3960 byte bei 4000 byte Seitenlänge (4-Kbyte-Seitenformat)
– 8056 byte bei 8096 byte Seitenlänge (8-Kbyte-Seitenformat)

Die maximale Länge eines Satzes mit variablem Feld kann jedoch in Abhängigkeit von der Verknüpfung des Satzes (siehe „SCD“ auf Seite 217) geringfügig kleiner sein.

Da der Satz außer dem variablen Feld mindestens noch das Satzlängenfeld *feldname-2* enthält, beträgt die maximale Länge des variablen Feldes:

– 2010 byte bei 2048 byte Seitenlänge
– 3958 byte bei 4000 byte Seitenlänge
– 8054 byte bei 8096 byte Seitenlänge

Enthält der Satz neben dem variablen Feld und dem Satzlängenfeld noch weitere Felder, so vermindert sich die maximale Länge des variablen Feldes entsprechend.

Außerdem gelten folgende Einschränkungen:

– Sie können pro Satzart höchstens ein Feld variabler Länge definieren. Es muss das letzte Feld einer Satzart sein.
– Ein Feld variabler Länge dürfen Sie nicht zum Schlüsselfeld erklären und nicht zum Direktzugriff auf Sätze verwenden.
– Eine Satzart, die ein Feld variabler Länge enthält, kann mit der Subschema-DDL nicht neu strukturiert werden. Sie muss genauso in das Subschema übernommen werden, wie Sie sie mit der Schema-DDL definieren.
– Eine Satzart, die ein Feld variabler Länge enthält, kann nicht mit SQL angesprochen werden.

Beispiel

```
RECORD NAME IS ARTIKELSTAMM
    DATENFELD-1            TYPE IS
        .                     .
        .                     .
        .                     .
    LAENGENFELD            TYPE IS BINARY 15.
    ARTIKELERLAEUTERUNG    PICTURE IS LX(500) DEPENDING ON LAENGENFELD.
```

4.2.6 Definieren eines nationalen Feldes (UTF-16)

Detaillierte Informationen finden Sie im COBOL2000-Handbuch „Sprachbeschreibung" unter „Zeichendarstellung durch UTF-16“.

```
[stufennummer ]feldname PICTURE IS maskenzeichenkette.
```

Nationale Felder können beliebige Zeichen aufnehmen.

Über *stufennummer* bestimmen Sie, ob das Feld einer Wiederholungsgruppe angehört:
Soll das Feld nicht zu einer Wiederholungsgruppe gehören, müssen Sie die Stufennummer so wählen, dass in der gesamten Satzart keine kleinere Stufennummer vorkommt.
Dies gilt insbesondere, wenn Sie das Feld als Schlüsselfeld benutzen wollen.
Soll das Feld Bestandteil einer Wiederholungsgruppe sein, gehen Sie gemäß Seite 62 vor.

Mit *feldname* geben Sie dem Feld einen Namen.

maskenzeichenkette darf aus folgenden Symbolen bestehen:

Symbol	Bezeichnung	Erläuterung
N		symbolisiert eine Speicherstelle, die jedes beliebige Zeichen des Zeichenvorrats aufnimmt.
(*ganzzahl*)	Wiederholungssymbol	Sie können das Symbol N wiederholen, indem Sie es mehrmals hintereinanderschreiben oder es nur einmal schreiben, gefolgt von dem Wiederholungssymbol: *ganzzahl* bezeichnet dabei den Wiederholungsfaktor. Standardwert ist 1; Maximalwert ist 127.

Tabelle 6: Symbole der *maskenzeichenkette* beim Definieren eines nationalen Feldes

Das erste Symbol in der Maskenzeichenkette muss N sein. Die Maskenzeichenkette darf nur aus höchstens 30 Zeichen bestehen.

Mit *ganzzahl* geben Sie die Anzahl der Speicherstellen des Feldes an, wobei jede Speicherstelle ein Zeichen des Zeichenvorrats aufnehmen kann.

Ein nationales Zeichen belegt 2 Bytes und wird in Datenstrukturen auf Bytegrenze ausgerichtet (siehe COBOL2000-Handbuch „Sprachbeschreibung" unter „Zeichendarstellung durch UTF-16“).

4.2.7 Definieren eines Database-Key-Feldes

```
[stufennummer ]feldname TYPE IS {DATABASE-KEY.        }
                                {DATABASE-KEY-LONG.   }
```

Database-Key-Felder sind binäre Felder, die für die Speicherung von Database-Key-Werten vorgesehen sind. Gleichzeitig sind sie die einzigen Felder, deren Inhalt UDS/SQL als Database-Key-Wert interpretiert.

Sie müssen ein Database-Key-Feld definieren, wenn die Database-Key-Werte nicht standardmäßig durch UDS/SQL, sondern vom Datenbankprogrammierer selbst vergeben werden sollen (siehe Abschnitt „Vergabe der Database-Key-Werte durch den Anwender“ auf Seite 81).

Über *stufennummer* bestimmen Sie, ob das Feld einer Wiederholungsgruppe angehört:
Soll das Feld nicht zu einer Wiederholungsgruppe gehören, müssen Sie die Stufennummer so wählen, dass in der gesamten Satzart keine kleinere Stufennummer vorkommt.
Dies gilt insbesondere, wenn Sie das Feld als Schlüsselfeld benutzen wollen.

Soll das Feld Bestandteil einer Wiederholungsgruppe sein, gehen Sie gemäß Seite 62 vor.

Ein Database-Key-Feld können Sie als Feld mit dem Datentyp DATABASE-KEY oder als Feld mit dem Datentyp DATABASE-KEY-LONG definieren:

– Ein DATABASE-KEY-Feld ist ein 4 byte langes binäres Feld mit dem Wertebereich 0 - 2^{31}-1.
– Ein DATABASE-KEY-LONG-Feld ist ein 8 byte langes binäres Feld mit dem Wertebereich 0 - 2^{63}-1, wobei die Bit-Positionen 17 - 32 (von links) von UDS/SQL nicht ausgewertet werden.

Der Aufbau von Database-Key-Werten ist auf Seite 128 näher beschrieben.

Ein Database-Key-Feld muss vom Datenbankprogrammierer mit Werten versorgt werden. Auf diese Weise *kann* der Datenbankprogrammierer den von UDS/SQL verwendeten Database-Key-Wert selbst festlegen (siehe LOCATION MODE-Klausel auf Seite 81).
Der Inhalt des Database-Key-Feldes *muss* aber nicht unbedingt mit dem von UDS/SQL intern benutzten Database-Key-Wert des Satzes übereinstimmen.

4.3 Definieren eines Vektors

```
[stufennummer ]vektorname { PICTURE.....  } OCCURS ganzzahl TIMES.
                          { TYPE.....     }
```

Ein Vektor ist ein Feld mit Wiederholungsfaktor. Der Wiederholungsfaktor muss größer als 1 sein. Er gibt an, wie viele Duplikate des Feldes zu dem Vektor zusammengefasst werden.

Sie definieren den Vektor wie ein Feld gemäß Seite 51.

Mit *ganzahl* geben Sie zusätzlich den Wiederholungsfaktor an.

Felder variabler Länge und Schlüsselfelder dürfen Sie nicht zu einem Vektor erklären.

Über *stufennummer* bestimmen Sie, ob der Vektor einer Wiederholungsgruppe angehört: Soll der Vektor nicht zu einer Wiederholungsgruppe gehören, müssen Sie die Stufennummer so wählen, dass in der gesamten Satzart keine kleinere Stufennummer vorkommt.

Wenn Sie den Vektor zum Bestandteil einer Wiederholungsgruppe erklären wollen, gehen Sie gemäß Seite 62 vor.

Die Begrenzung ist die maximale Satzlänge.

Beispiel

```
02 ADRESSE PICTURE IS X(20) OCCURS 2 TIMES.
```

4.4 Definieren einer Wiederholungsgruppe

```
[stufennummer-1 ]datengruppenname OCCURS ganzzahl TIMES.

{stufennummer-2 satzelementname[ {PICTURE.....}][ OCCURS.....].}...
                                 {TYPE.....   }
```

Eine Datengruppe ist eine benennbare Zusammenfassung von Satzelementen innerhalb einer Satzart, wobei jedes Satzelement ein Feld, ein Vektor oder selbst eine Datengruppe sein kann.

Eine Wiederholungsgruppe ist eine Datengruppe mit Wiederholungsfaktor. Der Wiederholungsfaktor muss größer als 1 sein. Er gibt an, wie viele Duplikate der Datengruppe zu der Wiederholungsgruppe zusammengefasst werden.

Die Definition von Datengruppen, die nicht Wiederholungsgruppen sind, ist nur für Subschemata sinnvoll und ist deshalb mit der Schema-DDL nicht möglich.

Mit *datengruppenname* geben Sie der Wiederholungsgruppe einen Namen.

Mit *ganzzahl* geben Sie den Wiederholungsfaktor an.

satzelementname bezeichnet ein Satzelement, das Sie zum Bestandteil der Wiederholungsgruppe erklären wollen. Sie definieren es gemäß Seite 51 (Feld) bzw. gemäß Seite 61 (Vektor) bzw. gemäß dieser Seite (Wiederholungsgruppe).

stufennummer-2 muss dabei größer sein als *stufennummer-1*.

Für alle Satzelemente, die Sie zum Bestandteil der Wiederholungsgruppe erklären, gilt außerdem:
Satzelemente müssen die gleiche Stufennummer haben, wenn ihnen die nächsthöhere Wiederholungsgruppe gemeinsam ist.

Wenn Sie eine Hierarchie von Wiederholungsgruppen definieren, darf diese höchstens dreistufig sein. Über einem Vektor dürfen Sie nur zwei Hierarchiestufen definieren.

Die Begrenzung ist die maximale Satzlänge.

Wenn Sie Elementarfelder einer Wiederholungsgruppe als Schlüsselfelder verwenden wollen, müssen Sie bedenken, dass die jeweils erste Ausprägung der übergeordneten Wiederholungsgruppe zugrunde gelegt wird und dies sich ggf. rekursiv bis zur äußeren Wiederholungsgruppe fortsetzt.

Beispiel

```
01  ANSCHRIFTEN OCCURS 2 TIMES.
    02 ADRESSE  PICTURE IS X(20) OCCURS 2 TIMES.
    02 TEL      PICTURE IS X(12).
```

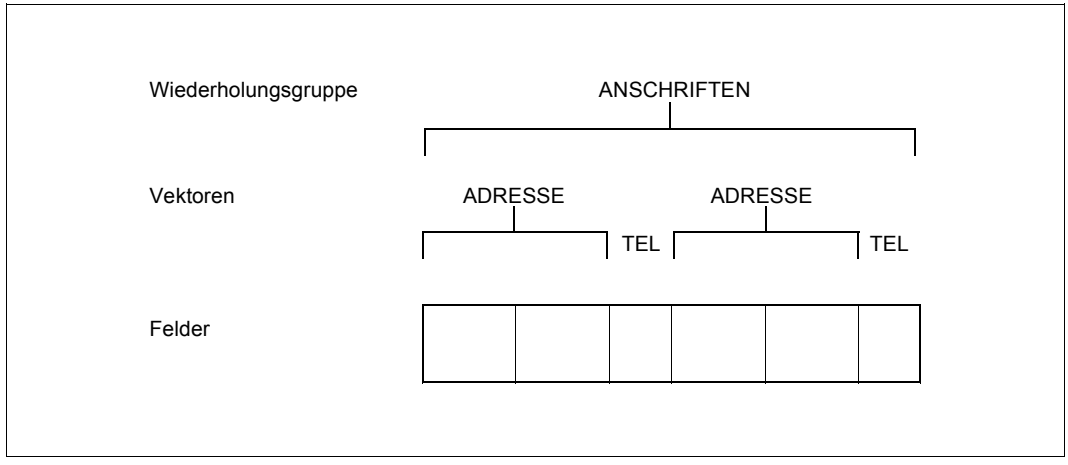


Bild 10: Zusammenfassung von Feldern und Vektoren zu einer Wiederholungsgruppe

4.5 Satzelemente zu einer Satzart zusammenfassen

```
RECORD NAME IS satzname
          .
          .
          .
   {[stufennummer ]satzelementname[ {PICTURE.....}][ OCCURS.....],}...
                                    {TYPE.....   }
```

Eine Satzart ist eine benennbare Zusammenfassung von Satzelementen, wobei jedes Satzelement ein Feld, ein Vektor oder eine Wiederholungsgruppe sein kann.

Eine einzelne Ausprägung einer Satzart heißt Satz. Ein Satz besteht somit aus je einem Feldinhalt aller am Aufbau der Satzart beteiligten Felder.

Eine Satzart ist außerdem die kleinste Dateneinheit, die UDS/SQL über einen eindeutigen Identifizierer, den Database Key, verwaltet. Deshalb müssen Sie Satzelemente als Bestandteile von Satzarten definieren.

Mit *satzname* vergeben Sie den Namen für die Satzart.

satzelementname bezeichnet ein Satzelement, das Sie zum Bestandteil der Satzart erklären. Sie definieren es gemäß den Seiten 51 bis 61.

Die Gesamtlänge aller Satzelemente innerhalb einer Satzart darf die maximale Satzlänge nicht überschreiten.
Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, beträgt die maximale Satzlänge:

- 2020 byte bei 2048 byte Seitenlänge (2-Kbyte-Seitenformat)
- 3968 byte bei 4000 byte Seitenlänge (4-Kbyte-Seitenformat)
- 8064 byte bei 8096 byte Seitenlänge (8-Kbyte-Seitenformat)

In Abhängigkeit von der Verknüpfung des Satzes (siehe „SCD“ auf Seite 217) kann die maximale Satzlänge auch geringfügig kleiner sein.

Für die maximale Anzahl Satzarten pro Datenbank gilt:

- Im Schema einer Datenbank mit 2048 byte Seitenlänge können Sie maximal 253 Satzarten definieren.
- Im Schema einer Datenbank mit 4000 byte oder 8096 byte Seitenlänge können Sie maximal 32 766 Satzarten definieren.

Beispiel

```
RECORD NAME IS KUNDE
          .
          .
          .
01  K-NR          PICTURE IS 9(10).
01  K-NAME        PICTURE IS X(20).
01  ANSCHRIFTEN   OCCURS 2 TIMES.
    02 ADRESSE    PICTURE IS X(20) OCCURS 2 TIMES.
    02 TEL        PICTURE IS X(12).
```

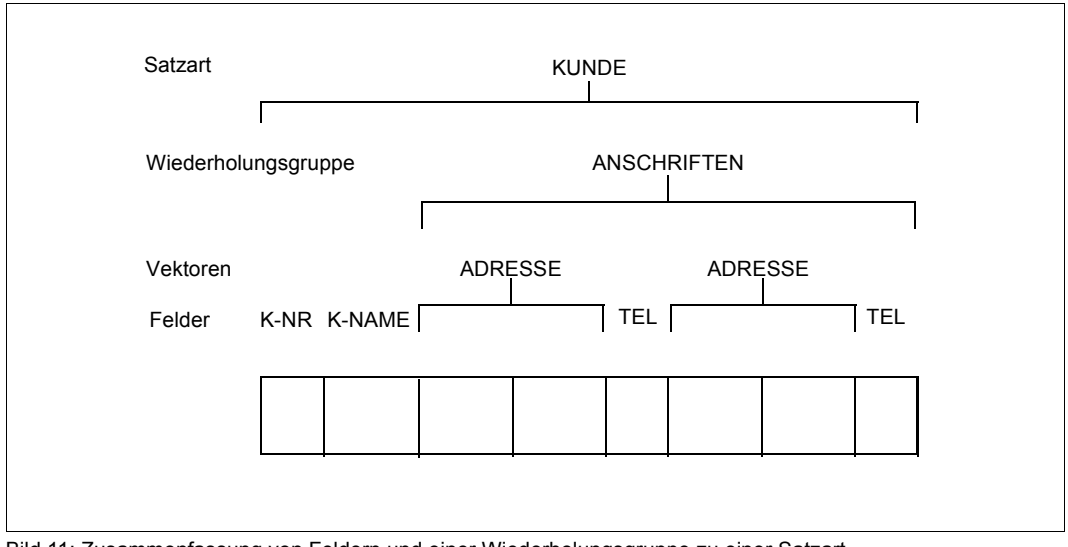


Bild 11: Zusammenfassung von Feldern und einer Wiederholungsgruppe zu einer Satzart

4.6 Sätze zweier Satzarten zu einem Set verbinden

Die Beziehung und Abhängigkeiten von Informationen innerhalb der Betriebsorganisation und der geplanten Datenbankanwendung werden in UDS/SQL als Verbindung zwischen Satzarten mit Hilfe des Setkonzepts nachgebildet.

Bei einer relationalen Anwendung erreichen Sie mit der Definition von Set-Beziehungen, dass die Fremdschlüssel entsprechend vergeben werden. Das ist die Voraussetzung für Verknüpfungen von Tabellen (Join) über implizite Set-Beziehungen.

Pro Datenbank können Sie maximal 32 766 Sets definieren.

Pro Satzart, die Owner eines Sets ist, dürfen Sie in diesen Sets in Summe maximal 255 Tabellen erzeugen; eine Tabelle wird angelegt, wenn der Set-Mode Pointer-Array oder List oder Chain sorted indexed ist, ebenso für jeden Sekundärschlüssel in diesen Sets.

Unabhängig davon dürfen Sie maximal 255 Sekundärschlüssel auf Satzartebene pro Satzart sowie auf Setebene pro singulärem Set definieren, wobei Hashverfahren nicht mitzählen.

4.6.1 Eine Set-Beziehung definieren

```
SET NAME IS setname
    .
    .
    .
    OWNER IS satzname-1.
MEMBER IS satzname-2.....
    .
    .
    .
```

Ein Set ist eine benennbare Beziehung zwischen zwei Satzarten. Diese Beziehung ist hierarchischer Natur, da Sie eine der beiden Satzarten zur übergeordneten Satzart, die andere zur untergeordneten Satzart erklären.

Die übergeordnete Satzart heißt Ownersatzart des Set.
Für *satzname-1* geben Sie den Namen der Satzart an, die Sie zur Ownersatzart erklären wollen.

Die untergeordnete Satzart heißt Membersatzart.
Mit *satzname-2* erklären Sie eine Satzart zur Membersatzart.

Mit *setname* geben Sie der Zusammenfassung beider Satzarten zu einem Set einen Namen.

Eine einzelne Ausprägung eines Sets heißt Set-Occurrence. Eine Set-Occurrence besteht aus genau einem Ownersatz und beliebig vielen ihm untergeordneten Membersätzen. Zu einem Set gehören also genau soviele Set-Occurrences wie Ownersätze vorhanden sind. Ist einem Ownersatz kein Membersatz zugeordnet, ist die Set-Occurrence leer.

Dargestellt werden Sets und Set-Occurrences nach folgendem Prinzip:

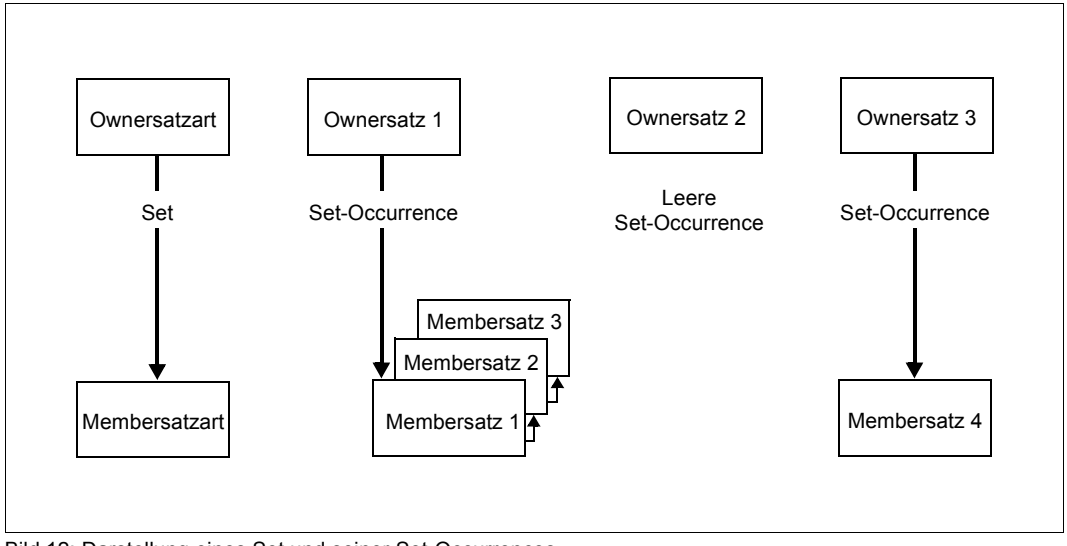


Bild 12: Darstellung eines Set und seiner Set-Occurrences

Bild 12 zeigt alle Sätze der Ownersatzart und somit alle Set-Occurrences des Set. Die Membersätze sind nur gezeigt, soweit sie Mitglied einer Set-Occurrence sind.

Innerhalb einer Set-Occurrence nehmen die Membersätze eine logische Reihenfolge ein. Die Reihenfolge ist durch die Pfeile der Membersätze auf den jeweiligen Nachfolger dargestellt.

Beispiel

```
RECORD NAME IS LIEFERANT
          .
          .
          .
RECORD NAME IS ARTIKEL
          .
          .
          .
SET NAME IS LIEFERBARE-ARTIKEL
          .
          .
          .
        OWNER IS LIEFERANT.
MEMBER IS ARTIKEL.....
          .
          .
          .
```

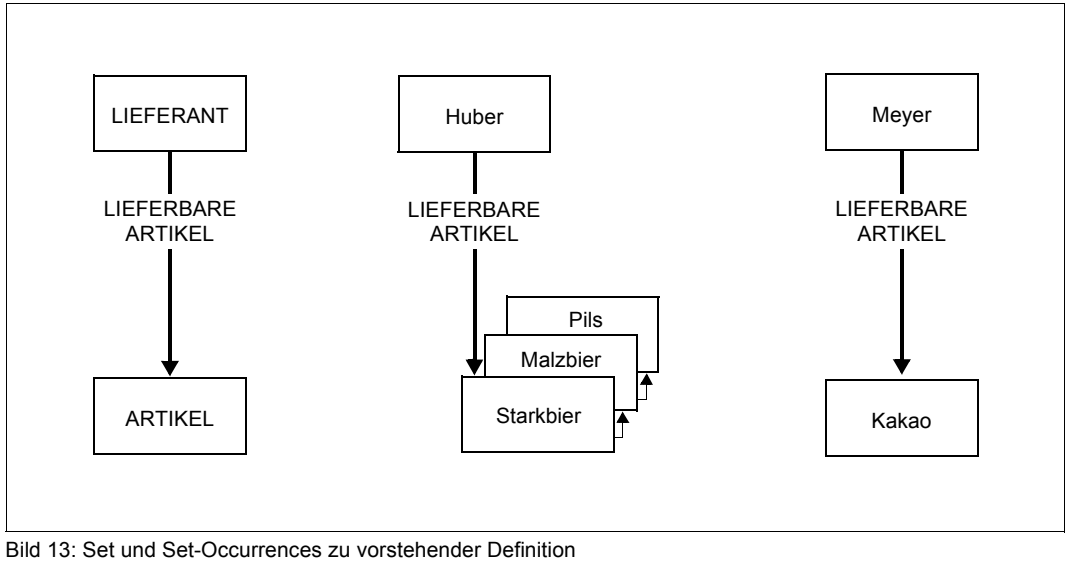


Bild 13: Set und Set-Occurrences zu vorstehender Definition

Eine Satzart kann in mehreren Sets enthalten sein. Damit ist der Set der elementare Baustein für netzartige Datenstrukturen.

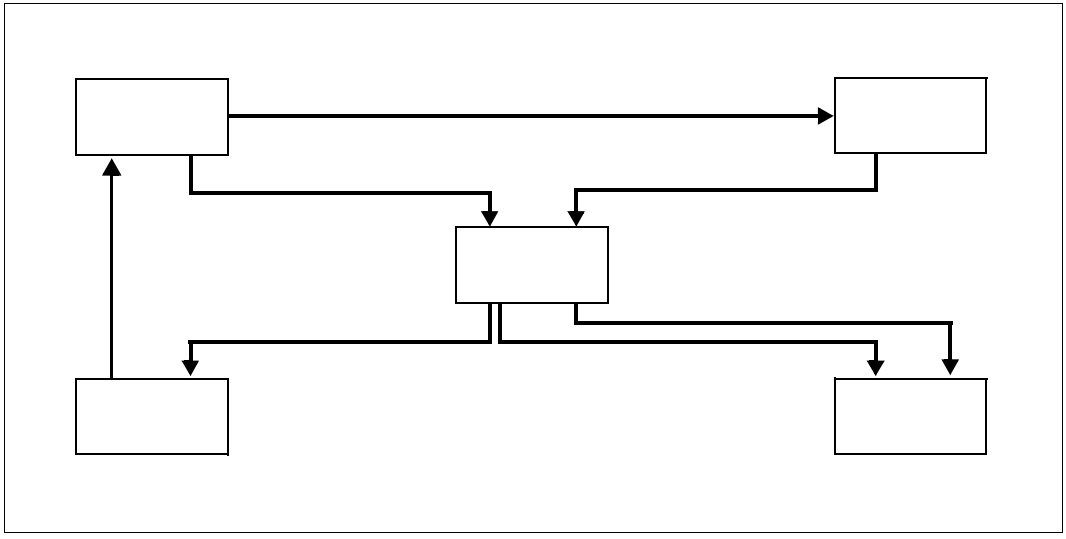


Bild 14: Netzartige Datenstruktur

Beziehungen zwischen Satzarten

Es gibt drei verschiedene Möglichkeiten, in denen Sätze der zu verknüpfenden Satzarten in Beziehung stehen können:

– 1:n-Beziehung zwischen zwei Satzarten
– m:n-Beziehung zwischen zwei Satzarten
– m:n-Beziehung innerhalb einer Satzart

1:n-Beziehung

Wenn Sie die Beziehung zwischen Sätzen durch einen Set darstellen möchten, müssen Sie folgende Regel beachten:
Ein Satz darf in jedem Set, in dem er Membersatz ist, nur höchstens einer Set-Occurrence angehören, d.h. in einem Set ist ein Membersatz höchstens einem Ownersatz zugeordnet. Zu einem Ownersatz können aber mehrere Membersätze gehören.

Es muss also eine 1:n-Beziehung zwischen Owner- und Membersatzart bestehen.

Beispiel für eine 1:n-Beziehung

Die Beziehung zwischen Kunden und ihren Aufträgen ist eine 1:n-Beziehung.
Ein Kunde kann mehrere Aufträge erteilen. Jeder Auftrag kann aber nur von einem Kunden gegeben werden.

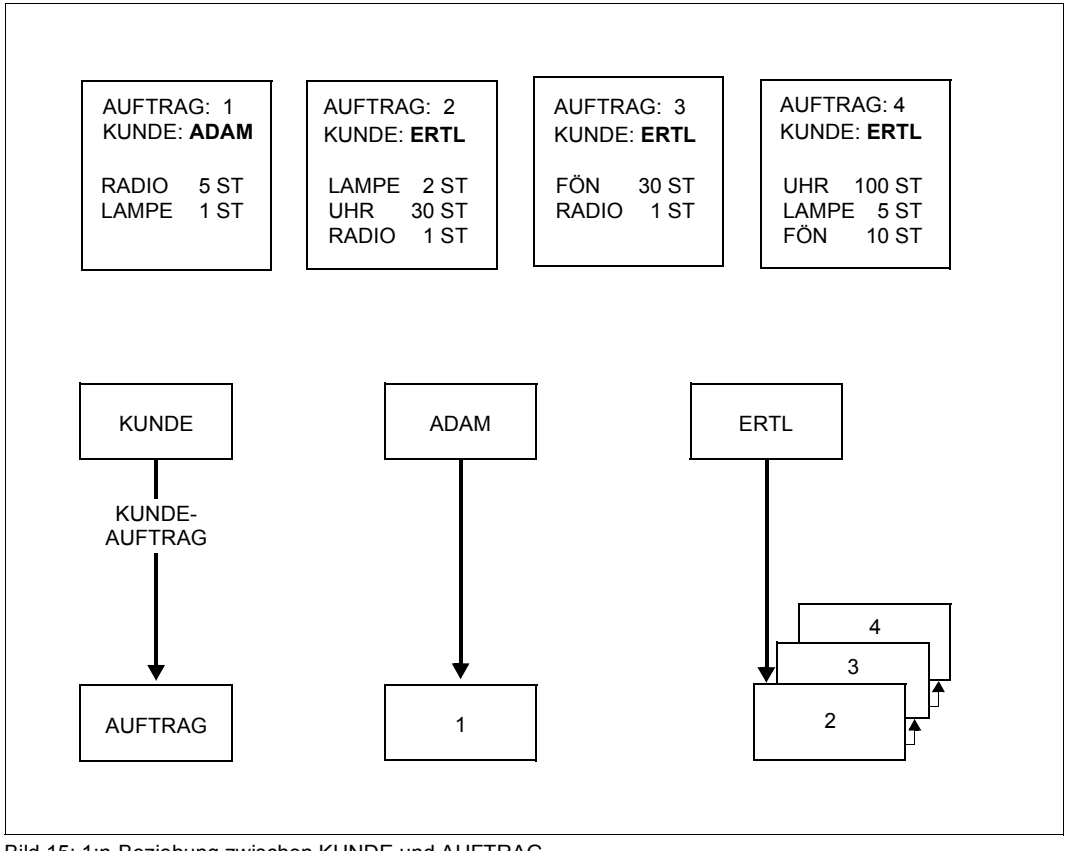


Bild 15: 1:n-Beziehung zwischen KUNDE und AUFTRAG

Die logische Beziehung wird durch die Definition eines Set KUNDE-AUFTRAG zwischen den betroffenen Satzarten hergestellt.

m:n-Beziehung zwischen zwei Satzarten (Vielfachbeziehung)

Eine m:n-Beziehung entsteht, wenn sich nicht jeder Membersatz mit nur einem Ownersatz verknüpfen lässt.

Wollen Sie eine m:n-Beziehung verwenden, geht dies durch das Auflösen in zwei einfache 1:n-Beziehungen. Sie definieren eine neue Satzart (Hilfssatzart), die dann als Membersatzart fungiert. Nach dem Auflösen hat jeder Membersatz wieder einen Owner.

Beispiel für eine m:n-Beziehung

Ein Beispiel für eine m:n-Beziehung ist die Beziehung zwischen Aufträgen und Artikel. In einem Auftrag können mehrere Artikel bestellt werden, ein Artikel kann in mehreren Aufträgen genannt sein.

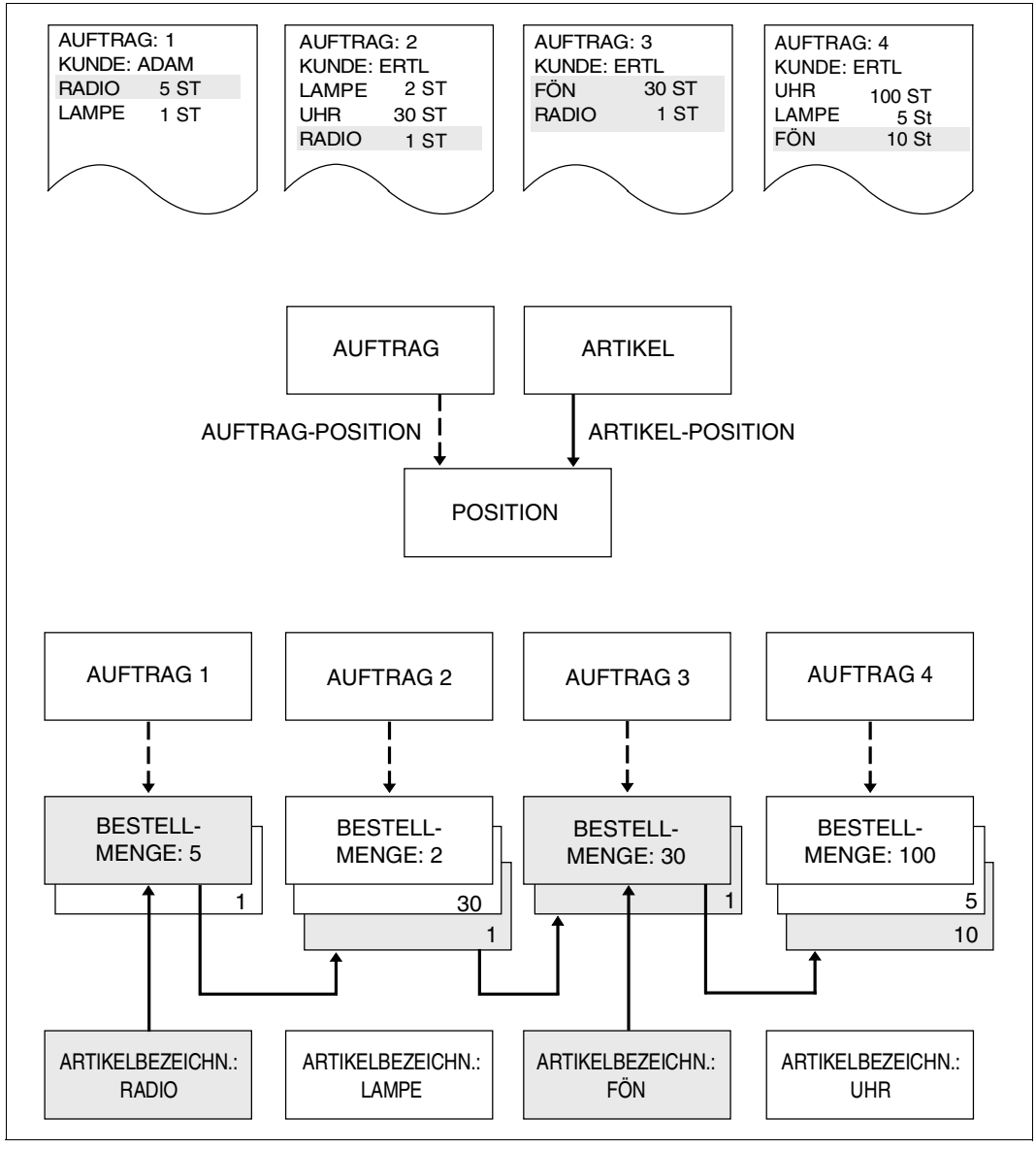


Bild 16: m:n-Beziehung zwischen AUFTRAG und ARTIKEL

Die m:n-Beziehung zwischen AUFTRAG und ARTIKEL wird aufgelöst in zwei einfache 1:n-Beziehungen, indem eine neue Satzart (Hilfssatzart) POSITION eingeführt wird, die Member von AUFTRAG und ARTIKEL ist.

m:n-Beziehung innerhalb einer Satzart (Stücklistenverarbeitung)

Die Besonderheit besteht darin, dass die m:n-Beziehung nicht zwischen zwei Satzarten besteht, sondern innerhalb einer Satzart.

Diese m:n-Beziehung wird aufgelöst, indem Sie eine neue Satzart (Hilfssatzart) definieren, die als Member fungiert, und zwei Sets definieren.

Beispiel für eine m:n-Beziehung innerhalb einer Satzart

Ein (Ober-)Teil besteht aus mehreren (Unter-)Teilen: → Stückliste

Ein (Unter-)Teil wird in mehreren (Ober-)Teilen verwendet: → Verwendungsnachweis

Ein Fahrrad besteht aus mehreren Teilen. Teile des Fahrrads werden wiederum in anderen Teilen des Fahrrades verwendet.

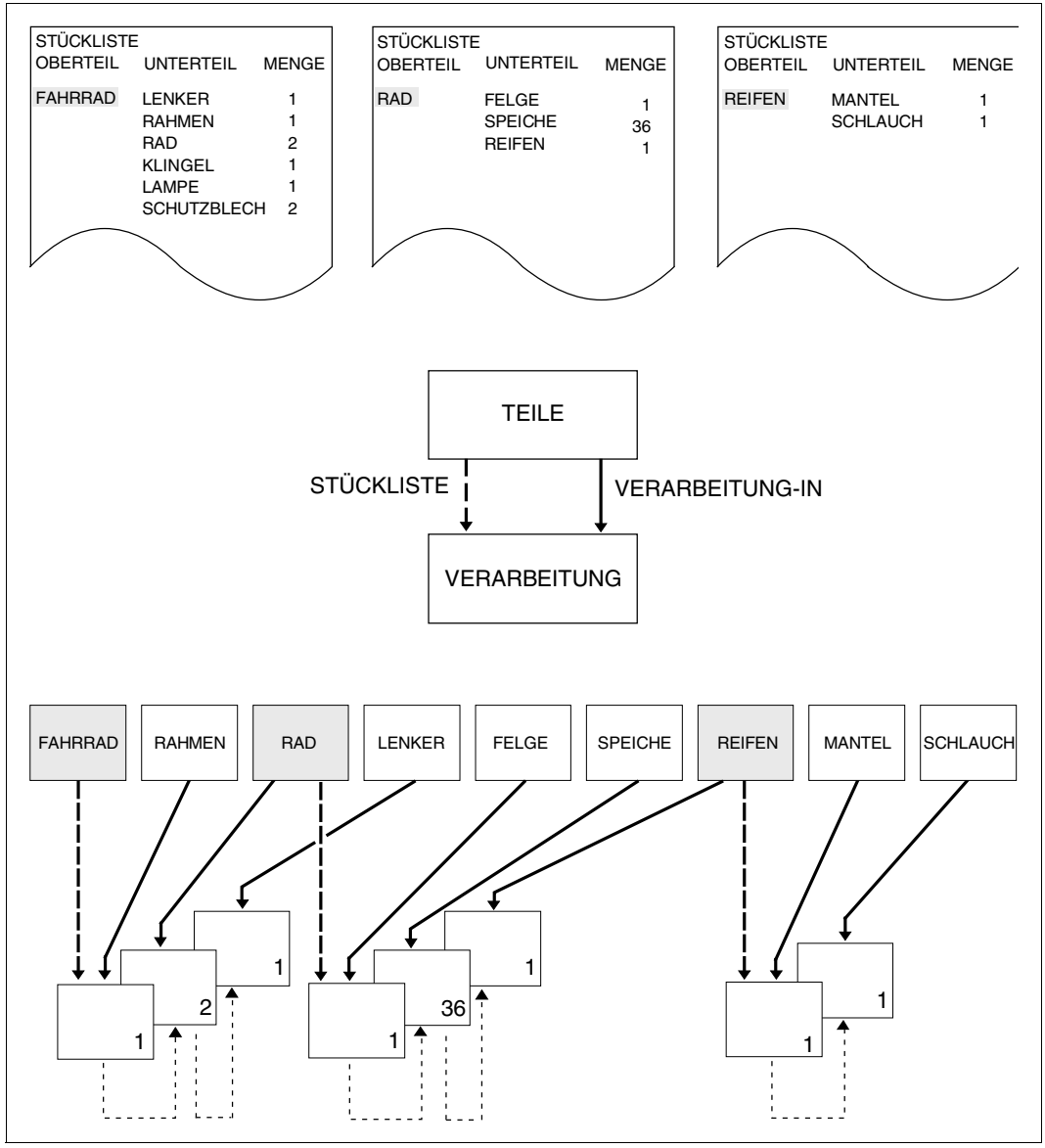


Bild 17: m:n-Beziehung innerhalb der Satzart TEILE

4.6.2 Art der Set-Mitgliedschaft von Membersätzen definieren

```
MEMBER IS satzname {MANDATORY} {AUTOMATIC}
                   {OPTIONAL } {MANUAL   }
```

Die Sätze einer Membersatzart müssen nicht automatisch Membersätze in einer Set-Occurrence sein. Die Mitgliedschaft eines Membersatzes in einer Set-Occurrence können Sie nach zwei Gesichtspunkten definieren:

1. Wann wird der Membersatz in die Set-Occurrence eingefügt?

   AUTOMATIC
   : Der Membersatz wird sofort beim Abspeichern automatisch in die Set-Occurrence eingefügt (siehe Handbuch „Anwendungen programmieren“, STORE).

   MANUAL
   : Der Membersatz wird beim Abspeichern nicht automatisch in die Set-Occurrence eingefügt. Die Mitgliedschaft in der Set-Occurrence wird erst durch eine spezielle DML-Anweisung hergestellt (siehe Handbuch „Anwendungen programmieren“, CONNECT).

2. Wie stark ist die Bindung eines bereits eingefügten Membersatzes an einen Ownersatz?

   MANDATORY
   : Die Verbindung ist starr. Die Existenz des Membersatzes ist nur sinnvoll in der Verbindung zu einem Ownersatz. In diesem Fall lässt sich die Mitgliedschaft in einer Set-Occurrence nur lösen, indem der Membersatz gleichzeitig in eine andere Set-Occurrence desselben Set überwechselt oder indem Sie ihn aus der Datenbank löschen (siehe Handbuch „Anwendungen programmieren“, MODIFY bzw. ERASE).

   OPTIONAL
   : Die Verbindung kann der Datenbankprogrammierer je nach Wunsch herstellen oder wieder aufheben (siehe Handbuch „Anwendungen programmieren“, CONNECT bzw. DISCONNECT).
   Insbesondere kann er die Verbindung aufheben, ohne den Membersatz zu löschen.

Beispiel 1

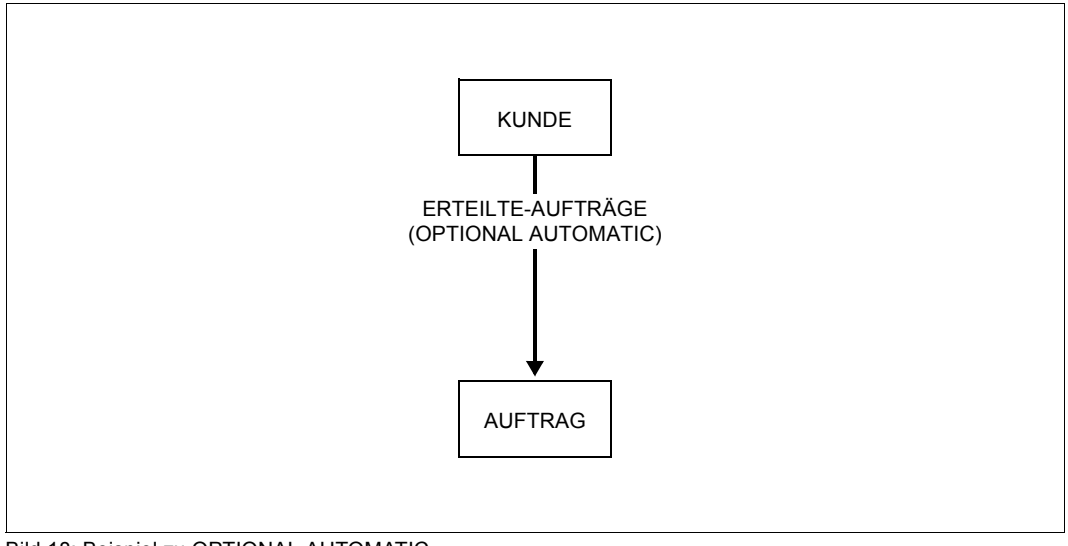


Bild 18: Beispiel zu OPTIONAL AUTOMATIC

Die Verbindung eines Auftragsatzes zu einem Kundensatz ist relativ starr: Es gibt erst einen Auftrag, wenn dieser von einem Kunden erteilt wurde. Also muss beim Einspeichern eines Auftragssatzes automatisch die Verbindung zu dem Ownersatz hergestellt werden. Erledigte Aufträge sollen für statistische Zwecke gespeichert bleiben, auch wenn es dann keine Verbindung mehr zu einem Kundensatz gibt. Deswegen ist die Set-Mitgliedschaft außerdem mit OPTIONAL zu definieren.

Beispiel 2

Von der MANUAL-Angabe machen Sie z.B. Gebrauch, wenn eine Satzart Membersatzart in zwei Sets ist, wobei bestimmte Membersätze jedoch nur einer Set-Occurrence angehören sollen.

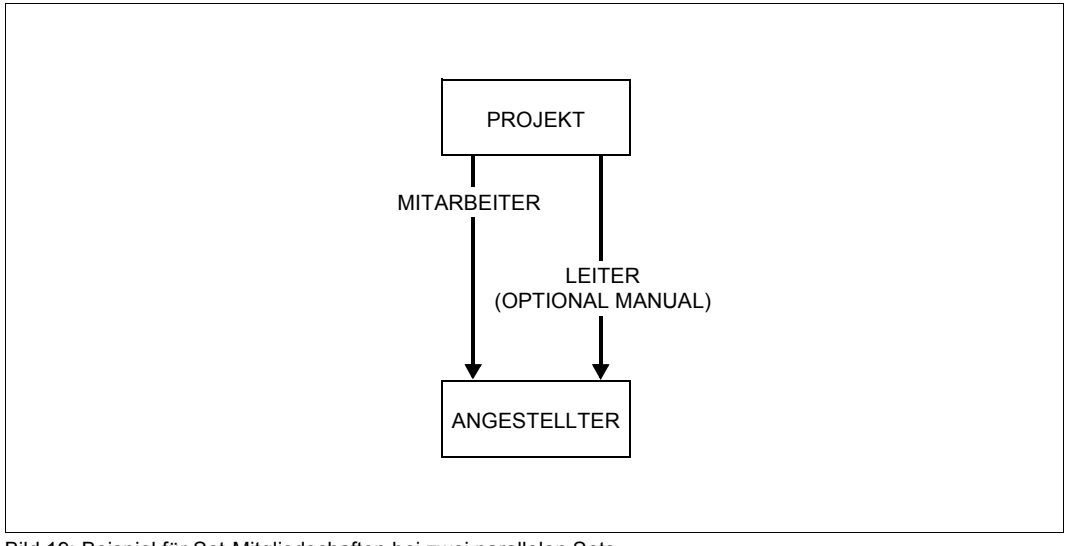


Bild 19: Beispiel für Set-Mitgliedschaften bei zwei parallelen Sets

Dieses Beispiel ist so gewählt, dass aus allen Angestellten, die an einem Projekt mitarbeiten, die Projektleiter über einen eigenen Set ausgewählt werden können. Da natürlich nicht alle Mitarbeiter eines Projekts Projektleiter sind, muss dieser Set mit MANUAL definiert werden. Erst wenn ein Angestellter die Leitung eines Projektes übernimmt, wird er in die zugehörige Set-Occurrence aufgenommen. Damit er aus der Set-Occurrence gelöscht werden kann, wenn er die Leitung des Projektes wieder abgibt, ist der Set außerdem mit OPTIONAL zu definieren.

Beispiel 3

Sie benötigen die MANUAL-Angabe, wenn Sie in Ihrer Datenstruktur einen Zyklus definiert haben, d.h. wenn Sie eine Anzahl von Satzarten so miteinander verknüpft haben, dass jede Satzart in einem Set Owner und gleichzeitig in einem anderen Set Member ist.

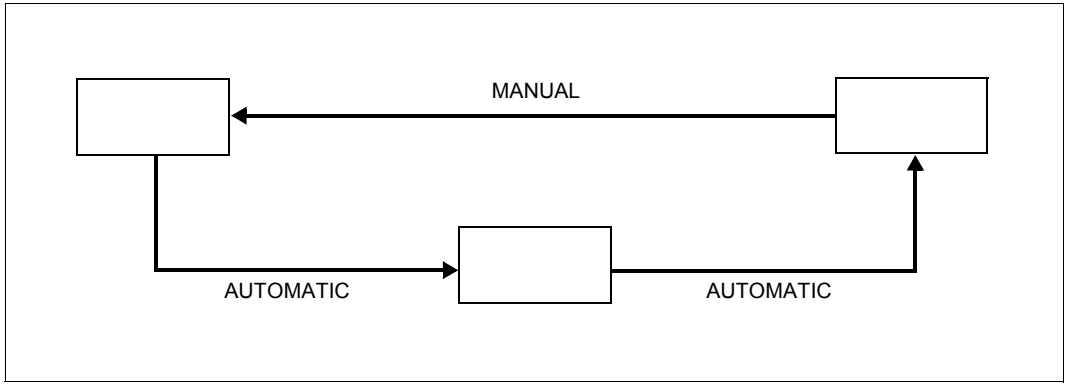


Bild 20: Art der Set-Mitgliedschaft in einem Zyklus

Würden Sie alle Sets eines Zyklus mit AUTOMATIC definieren, wäre es unmöglich einen Satz dieses Zyklus in die Datenbank einzuspeichern, da die AUTOMATIC-Angabe immer voraussetzt, dass beim Einspeichern eines Satzes dessen Ownersatz schon vorhanden ist.

Beispiel 4

Um eine Vielfachbeziehung aufzulösen, definieren Sie eine Hilfssatzart. Dazu sind folgende Set-Mitgliedschaften sinnvoll:

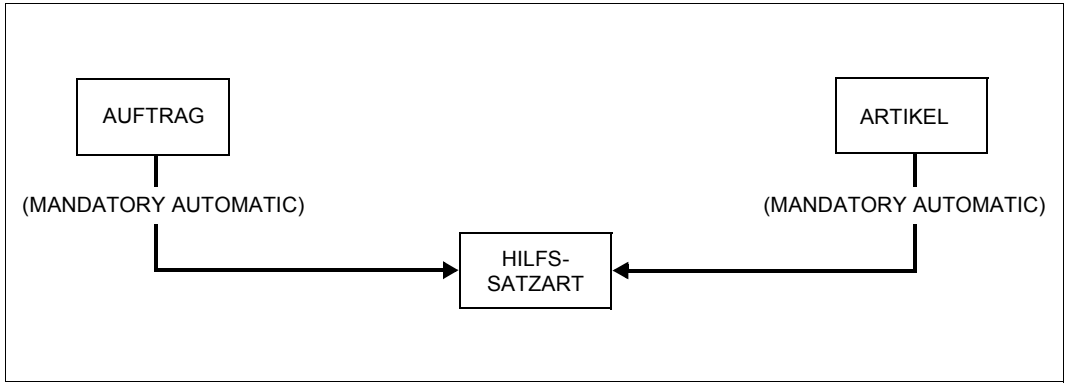


Bild 21: Set-Mitgliedschaften bei Vielfachbeziehungen

Der Sinn der Hilfssatzart ist es, die Auftragssätze und Artikelsätze miteinander zu verknüpfen. Die Verknüpfung wird nur hergestellt, wenn jeder Hilfssatz Mitglied je einer Set-Occurrence beider Sets ist. Deshalb kann die Mitgliedschaft mit AUTOMATIC erzwungen werden.

Wenn die Informationen in den Hilfssätzen nur in Verbindung mit dem Inhalt der Ownersätze aussagekräftig sind, ist es sinnlos, die Verbindung zwischen Hilfssatz und Ownersatz zu lösen. Sinnvoll ist nur ein Löschen der Sätze. Durch MANDATORY erreichen Sie eine starre Verbindung zwischen den Hilfssätzen und den Ownersätzen.

4.7 Zugriffspfade und Satzreihenfolgen

Sie können mit der DDL Einfluss nehmen auf folgende Zugriffsarten:

– direkter Zugriff auf Satzartebene

  Ein Satz wird über den Inhalt eines Feldes bzw. einer Kombination von Feldern aus der Menge aller Sätze einer Satzart ausgewählt.

– sequenzieller Zugriff auf Satzartebene

  Ein Satz wird auf Grund der Position ausgewählt, die er innerhalb der logischen Reihenfolge aller Sätze einer Satzart einnimmt.

– direkter Zugriff auf Setebene

  Ein Satz wird über den Inhalt eines Feldes bzw. einer Kombination von Feldern aus der Menge aller Sätze einer Set-Occurrence ausgewählt.

– sequenzieller Zugriff auf Setebene

  Ein Satz wird auf Grund der Position ausgewählt, die er innerhalb der logischen Reihenfolge aller Sätze einer Set-Occurrence einnimmt.

Im Folgenden ist beschrieben,

– welche Zugriffspfade für direkte Zugriffe standardmäßig vorhanden sind und welche Sie zusätzlich einrichten können.

– wie Sie die logische Reihenfolge von Sätzen für die gewünschte sequenzielle Verarbeitung definieren.

4.7.1 Direkter und sequenzieller Zugriff auf Satzartebene über Database-Key-Wert

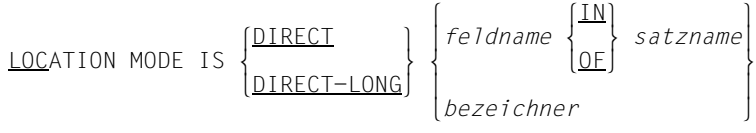
Der Database-Key-Wert ist ein eindeutiger interner Satzschlüssel, der beim Speichern eines Satzes vergeben wird und der während der gesamten Lebensdauer des Satzes nicht geändert werden kann.
Ein Database-Key-Wert setzt sich zusammen aus einem Kennzeichen für die Satzart, der Satzartnummer, und einer Satzfolgenummer (siehe Abschnitt „Aufbau eines Database-Key-Wertes“ auf Seite 128). Die Reihenfolge der Sätze innerhalb einer Satzart ist durch aufsteigende Satzfolgenummern bestimmt.

UDS/SQL legt für jede Satzart automatisch eine Tabelle, die Database Key Translation Table (DBTT) an, die die physischen Adressen (Seitennummern) aller Sätze der Satzart enthält (siehe Abschnitt „DBTT (Database Key Translation Table)“ auf Seite 127).
UDS/SQL findet die physische Adresse eines Satzes in der DBTT über die Umrechnung seines Database-Key-Wertes und muss die DBTT deshalb nicht sequenziell absuchen.

Über den Database Key sind somit immer direkte und sequenzielle Zugriffe auf Satzartebene möglich.

Standardmäßig werden die Database-Key-Werte von UDS/SQL vergeben. In der Regel ist dann die Reihenfolge der Sätze für eine sequenzielle Verarbeitung nicht vorhersehbar. Indem der Datenbankprogrammierer die Database-Key-Werte selbst einspeichert, kann er die Satzreihenfolge bestimmen. Dazu müssen Sie folgende Vorbereitungen treffen.

Vergabe der Database-Key-Werte durch den Anwender



Durch Angabe dieser Klausel geben Sie nicht nur dem Datenbankprogrammierer die Möglichkeit, die Reihenfolge der Sätze zu bestimmen, sondern Sie schaffen auch die Voraussetzung für eine komfortable Auswahlmethode der evtl. zugehörigen Set-Occurrences (siehe Abschnitt „Auswahlmethode für Set-Occurrences bestimmen“ auf Seite 98). Wenn Sie diese Klausel angeben, können Sie mit SQL keinen neuen Satz der angegebenen Satzart einfügen.

Mit *feldname* nennen Sie ein Feld, das Database-Key-Werte aufnehmen kann. Dieses erklären Sie dadurch auch zum Schlüsselfeld für direkten Zugriff.

Wenn Sie LOCATION MODE IS DIRECT angeben, müssen Sie *feldname* als DATABASE-KEY-Feld definieren.

Wenn Sie LOCATION MODE IS DIRECT-LONG angeben, müssen Sie *feldname* als DATABASE-KEY-LONG-Feld definieren.

Mit *satzname* bezeichnen Sie die Satzart, in der das Database-Key-Feld *feldname* enthalten ist.

Mit *bezeichner* vergeben Sie den Namen für ein Feld, das UDS/SQL automatisch zur Aufnahme von Database-Key-Werten erzeugt. Dieses Feld dient als Schlüsselfeld, ist aber nicht Bestandteil einer Satzart.
Wenn Sie LOCATION MODE IS DIRECT angeben, erzeugt UDS/SQL das Feld *bezeichner* als DATABASE-KEY-Feld.
Wenn Sie LOCATION MODE IS DIRECT-LONG angeben, erzeugt UDS/SQL das Feld *bezeichner* als DATABASE-KEY-LONG-Feld.

Zum Einspeichern eines Satzes kann der Datenbankprogrammierer das Feld *feldname* bzw. *bezeichner* mit dem Database-Key-Wert versorgen, der die Position des Satzes in der Satzreihenfolge angibt (siehe Abschnitt „DBTT (Database Key Translation Table)“ auf Seite 127).

Wenn der angegebene Database-Key-Wert belegt ist, vergibt UDS/SQL automatisch einen bislang noch nicht verwendeten Database-Key-Wert.

Wenn der Datenbankprogrammierer den Database-Key-Wert nicht selbst vergeben will, kann er stattdessen 0 angeben. Damit erreicht er wieder die automatische Vergabe der Database-Key-Werte durch UDS/SQL.

Nähere Informationen zur Vergabe eines Database Keys beim Speichern eines Satzes finden sie im Handbuch „Anwendungen programmieren“ im Nachschageteil der COBOL-DML unter „STORE, Database-Key-Wert zuordnen“.

4.7.2 Zusätzliche Zugriffspfade für Direktzugriff auf Satzartebene anlegen

Primärschlüssel zur Umrechnung durch Hashverfahren definieren

```
LOCATION MODE IS CALC[ hashroutine]
    USING feldname,... DUPLICATES ARE[ NOT] ALLOWED
```

Mit dieser Klausel entscheiden Sie sich für die gestreute Speicherung der Sätze über ein Hashverfahren. Der Speicherbereich, den Sie über dieses Verfahren adressieren, heißt Hashbereich.
Außerdem schaffen Sie die Voraussetzung für eine komfortable Auswahlmethode der zugehörigen Set-Occurrences, falls die Sätze Owner in einem Set sind (siehe Abschnitt „Auswahlmethode für Set-Occurrences bestimmen“ auf Seite 98).

Ein Schlüssel, den Sie mit LOCATION MODE IS CALC erklären, heißt CALC-Key. Er ist der Primärschlüssel für die Satzart. Er kann aus mehreren Feldern zusammengesetzt sein (Compound Key).

Mit *feldname* nennen Sie die Felder, aus denen Sie den Schlüssel zusammensetzen, wobei alle genannten Felder Teil der zugehörigen Satzart sein müssen.
Die Felder dürfen sowohl numerisch als auch alphanumerisch sein.

Mit DUPLICATES ARE[NOT] ALLOWED entscheiden Sie, ob UDS/SQL - bezogen auf einen Realm - gleiche Schlüsselwerte zurückweisen oder zulassen soll.

Mit *hashroutine* geben Sie den Namen eines Moduls an, wenn Sie die Lage der Daten im Hashbereich selbst steuern wollen. Machen Sie zu *hashroutine* keine Angabe, so verwendet UDS/SQL sein Standard-Hashverfahren.
Haben Sie eine *hashroutine* angegeben, so muss diese Ihren jeweiligen Primärschlüssel in eine vier byte lange Binärzahl umrechnen. Die Binärzahl setzt UDS/SQL anschließend in eine relative Seitennummer um. In dieser Seite befindet sich im Allgemeinen der zugehörige Satz (siehe Abschnitt „Direkte CALC-Seite“ auf Seite 208).

Wenn Sie eine eigene Hashroutine definieren wollen, müssen Sie folgende Registerkonventionen beachten:

- Mit Ausnahme von Register 1 müssen alle UDS/SQL-Register vor und nach dem Durchlaufen der Routine denselben Inhalt haben.
  Der Ansprung der Hashroutine erfolgt mit BALR 14,15; Rücksprung zur UDS/SQL-Hashroutine mit BR 14.
- Der UDS/SQL-DBH übergibt folgende Informationen:

  Register 1: adressiert ein Wort, das die Adresse des Schlüsselfeldes enthält.

  Register 2: adressiert ein Byte, das die Länge des Schlüsselfeldes enthält.

  Register 3: adressiert ein Wort, das die Anzahl der Seiten des Hashbereiches enthält, der diesem Schlüssel zugeordnet ist.

  Register 13: Ab der Adresse (Register 13)+X´0C´ stehen Ihnen 13 Wörter als transaktionsbezogener Sicherstellungsbereich für Register zur Verfügung.

- Der UDS/SQL-DBH erwartet das Ergebnis der Hashroutine im Register 1.

  Die im Register 1 enthaltene Binärzahl setzt UDS/SQL mit folgenden Operationen in eine relative Seitennummer um:

  – Das Bit 2^{31} im Register 1 wird auf null gesetzt.
  – Ist der sich ergebende Wert kleiner als die Anzahl der Seiten des Hashbereichs (Inhalt von Register 3), so wird dieser Wert als relative Seitennummer benutzt.
  – Ist der sich ergebende Wert größer, so führt UDS/SQL folgende Division aus:
    Inhalt (Register 1) / Inhalt (Register 3)

    Der sich aus dieser Rechnung ergebende Rest wird als relative Seitennummer benutzt.

Beispiel 1

Dieses Beispiel zeigt die Umrechnung eines Schlüsselwertes in eine physische Adresse mit dem Standard-Hashverfahren des UDS/SQL.

Der Schlüsselwert sei 9952333, die Anzahl der Seiten des Hashbereichs sei 503. Der Schlüsselwert durchläuft folgende Operationen:

```
1)  Aufteilung des          9952/333
    Schlüsselwertes von
    links beginnend in
    Wörter (Einheiten
    von vier byte Länge)

2)  Verknüpfen der zuge-    F9F9F5F2 ——> 11111001 11111001 11110101 11110010
    hörigen Binärdar-       00F3F3F3 ——> 00000000 11110011 11110011 11110011
    stellung durch ex-                   -----------------------------------
    klusives ODER                        11111001 00001010 00000110 00000001

3)  1. Bit auf 0 (=posi-       01111001 00001010 00000110 00000001=790A0601
    tiv) setzen

4)  Durch Anzahl der        790A0601(16) : 503(10)
    Seiten des Hashbe-
    bereichs teilen         =2030700033(10) : 503(10) = 4037117 Rest= 2
```

Der verbleibende Rest gibt die physische Adresse in Form einer relativen Seitennummer (siehe Abschnitt „Aufbau einer physischen Seitenadresse“ auf Seite 127) innerhalb des Hashbereichs an.

Durch eine eigene Hashroutine können Sie die ersten beiden Operationen ersetzen. Die beiden letzten Operationen führt UDS/SQL immer aus.

Beispiel 2

Dies ist ein Beispiel für eine Hashroutine, die Sie selbst programmieren können. Das Programm ersetzt die beiden ersten Operationen der Standard-Hashroutine des UDS/SQL durch einen Divisions-Rest-Algorithmus, der den gesamten Schlüsselwert als positive Binärzahl betrachtet und durch die Anzahl der zur Verfügung stehenden CALC-Seiten dividiert.

Der Algorithmus entspricht einer handschriftlich ausgeführten Division mit dem einzigen Unterschied, dass jeweils drei Stellen nachgezogen werden. Nachfolgend ein Beispiel zu dieser Vorgehensweise, der Einfachheit halber mit Dezimalzahlen:

```
1234567890 : 13
117
  6
  6456
  8
  8789
     1
     10
```

Der sich ergebende Rest ist die relative Seitennummer. Er ist immer kleiner als die Anzahl der CALC-Seiten, d.h. nach dem Rücksprung zur Standard-Hashroutine wird diese Seitennummer nicht mehr verändert. Der DBH übernimmt den im Register 1 übergebenen Binärwert als Endresultat.

Diese Hashroutine erzeugt die gleichen Ergebnisse wie die Standard-Hashroutine, wenn der Schlüsselwert nicht länger als vier byte ist.

Die Hashroutine BYTEHASH erzeugt aus einem CALC-Key logisch gesehen in 2 Schritten eine CALC-Seitennummer:

1. Die Reihenfolge der Bytes innerhalb des CALC-Key wird umgekehrt.
2. Der dadurch entstandene Bytestring wird in voller Länge als positive Ganzzahl behandelt und durch die Anzahl der CALC-Seiten dividiert. Der sich ergebende Divisionsrest ist dann die CALC-Seitennummer.

Beispiel in Assembler

```
BYTEHASH CSECT READ
BYTEHASH AMODE ANY
BYTEHASH RMODE ANY
         USING *,15
         STM   4,8,12(13)
         LM    4,7 ALLZEROS
         L     8,0(0,1)
         LA    4,0(0,8)
         BCTR  4,0      ---------------- (1)
         IC    5,0(0,2) ---------------- (2)
         DR    6,5      ---------------- (3)
HASHBYTE SRDL  6,24
         IC    7,0(4,5)
         D     6,0(0,3)
         BCT   5,HASHBYTE
         LR    1,6
         LM    4,8,12(13)
         BR    14
ALLZEROS DC    4F'0'
         END
```

(1) Register 4 enthält konstant die Adresse vor dem CALC-Key.

(2) Register 5 enthält den Index auf den CALC-Key, Anfangswert ist die Länge des CALC-Key.

(3) Diese Division führt zu einem P104, falls als Länge des CALC-Key eine Null übergeben wird.

Sekundärschlüssel zur Umrechnung durch Hashverfahren definieren

```
SEARCH KEY IS feldname,... USING CALC[ hashroutine]
    DUPLICATES ARE[ NOT] ALLOWED
```

Ein Schlüssel, den Sie mit SEARCH KEY IS... erklären, heißt SEARCH-Key oder auch Sekundärschlüssel. Der Schlüssel kann aus mehreren Feldern zusammengesetzt sein (Compound Key).

Mit *feldname* nennen Sie die Felder, aus denen Sie den Schlüssel zusammensetzen, wobei alle genannten Felder Teil der zugehörigen Satzart sein müssen.

Mit DUPLICATES ARE[NOT] ALLOWED entscheiden Sie, ob UDS/SQL gleiche Schlüsselwerte zurückweisen oder zulassen soll.

Mit *hashroutine* geben Sie den Namen eines Moduls an, der Ihren Sekundärschlüssel in eine vier byte lange Binärzahl umrechnet. Die Binärzahl setzt UDS/SQL anschließend in eine relative Seitennummer um. In dieser Seite findet UDS/SQL einen Zeiger auf den Satz (siehe Abschnitt „Indirekte CALC-Seite“ auf Seite 211).
Die Hashroutine berechnet nicht direkt die Satzadresse, weil es Ihnen vorbehalten ist, die Sätze entweder schon über den Primärschlüssel oder mit SSL-Anweisungen zu platzieren.

Wenn Sie Hashbereiche mit SSL-Anweisungen platzieren, müssen Sie für den Hash-Bereich einen Namen vergeben (siehe Seite 102).

Wenn Sie zu *hashroutine* keine Angabe machen, verwendet UDS/SQL dieselbe Standard-Hashroutine wie zur Umrechnung des Primärschlüssels (Zur Programmierung einer Hashroutine und zum Ablauf der Standard-Hashroutine siehe Seite 83 „Primärschlüssel zur Umrechnung durch Hashverfahren definieren“).

Sie können für eine Satzart mehrere Sekundärschlüssel definieren.

Beispiel

```
RECORD NAME IS ARTIKEL
          .
          .
          .
    SEARCH KEY IS ART-NR-LIEFER, FARB-NR-LIEFER, GROESSE USING CALC.....
    SEARCH KEY IS BEZEICHNUNG USING CALC.....
01  ART-NR-LIEFER                  PICTURE IS 9(4).
01  FARB-NR-LIEFER                 PICTURE IS 99.
01  GROESSE                        PICTURE IS 99.
01  BEZEICHNUNG                    TYPE IS CHARACTER 40.
```

Sekundärschlüssel für Direktzugriff über Tabelle definieren

```
SEARCH KEY IS feldname,... USING INDEX [NAME IS name]
              DUPLICATES ARE[ NOT] ALLOWED
```

Ein Schlüssel, den Sie mit SEARCH-KEY IS... erklären, heißt SEARCH-Key oder auch Sekundärschlüssel. Der Schlüssel kann aus mehreren Feldern zusammengesetzt sein (Compound Key).

Mit *feldname* nennen Sie die Felder, aus denen Sie den Schlüssel zusammensetzen, wobei alle genannten Felder Teil der zugehörigen Satzart sein müssen.

Mit *name* geben Sie den Namen der Tabelle an. In den die Tabelle betreffenden SSL-Anweisungen beziehen Sie sich auf diesen Namen.

Mit DUPLICATES ARE[NOT] ALLOWED entscheiden Sie, ob UDS/SQL gleiche Schlüsselwerte zurückweisen oder zulassen soll.

Auf Grund dieser Definition richtet UDS/SQL eine Satz-SEARCH-Key-Tabelle ein. Diese Tabelle nimmt die Werte des Sekundärschlüssels aller Sätze der Satzart auf und stellt eine eindeutige Beziehung zwischen dem Schlüsselwert, dem Database-Key-Wert und der physischen Adresse des Satzes her. Die physische Adresse wird jedoch bei einer Lageänderung des Satzes nicht automatisch aktualisiert.
Eine SEARCH-Key-Tabelle dient ausschließlich dem Direktzugriff auf die Sätze der Satzart. Sie wird immer mehrstufig angelegt, um den Direktzugriff zu beschleunigen.

Sie können unabhängig voneinander mehrere Sekundärschlüssel für eine Satzart definieren.

4.7.3 Reihenfolge von Sätzen innerhalb der Set-Occurrences festlegen

Wenn Sie die logische Reihenfolge bestimmen, die die Membersätze innerhalb der Set-Occurrences eines Set einnehmen sollen, haben Sie die Wahl zwischen zwei Kategorien:

– Reihenfolge ohne Sortierung nach den Werten eines Schlüssels
– Sortierung der Membersätze nach den Werten des Primärschlüssels

Sie sind im Folgenden genauer beschrieben.

Reihenfolge ohne Sortierung nach Schlüsselwerten

```
              ⎧LAST     ⎫
              ⎪         ⎪
              ⎪FIRST    ⎪
ORDER IS      ⎨NEXT     ⎬
              ⎪         ⎪
              ⎪PRIOR    ⎪
              ⎪         ⎪
              ⎩IMMATERIAL⎭
```

ORDER IS LAST

Hierbei nehmen die Membersätze in den Set-Occurrences die Reihenfolge ein, die der zeitlichen Reihenfolge des Einspeicherns der Sätze entspricht.

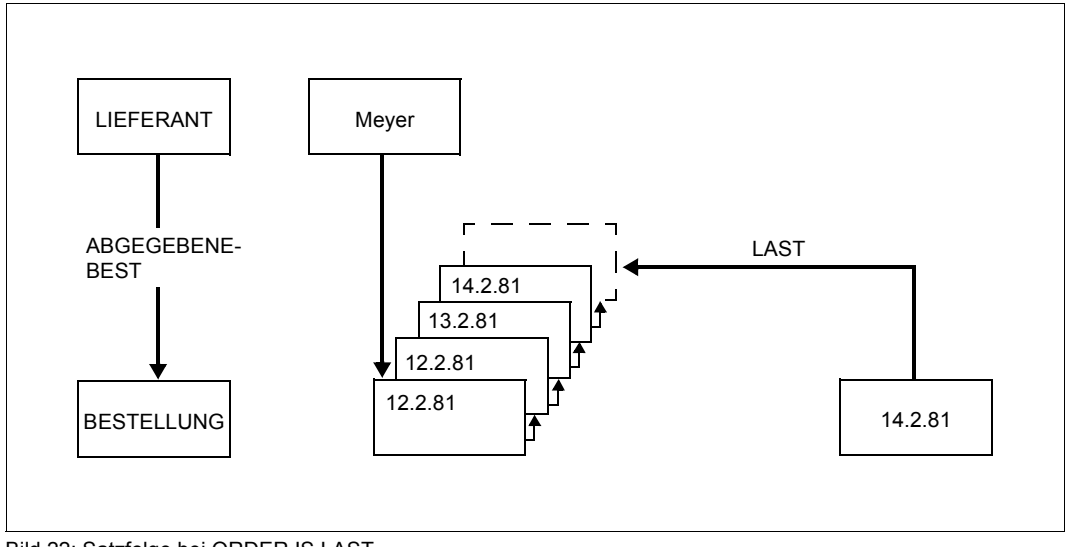


Bild 22: Satzfolge bei ORDER IS LAST

ORDER IS FIRST

Hiermit definieren Sie eine Ordnung, die der zeitlichen Reihenfolge des Einspeicherns der Membersätze entgegengesetzt ist.

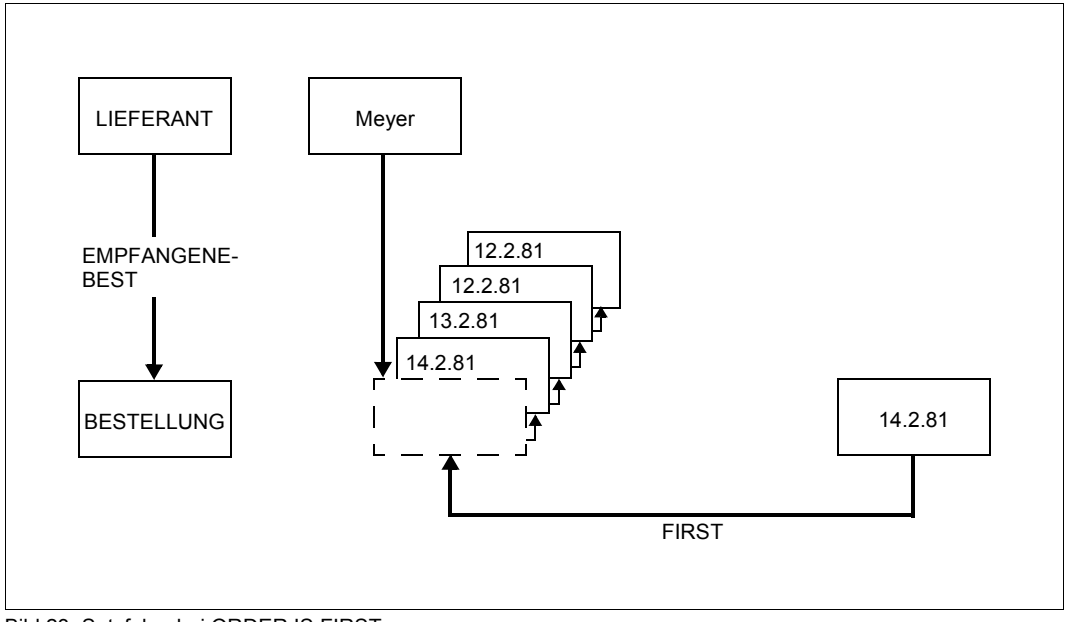


Bild 23: Satzfolge bei ORDER IS FIRST

ORDER IS NEXT/PRIOR

Hiermit geben Sie dem Datenbankprogrammierer die Möglichkeit, beim Einspeichern der Membersätze selbst eine bestimmte Reihenfolge herzustellen.
Der von ihm zuletzt angesprochene Satz eines Set (CURRENT RECORD OF SET oder CRS) gilt als Bezugspunkt für den Eintragungsort des einzuspeichernden Satzes. Der CRS bestimmt also

– die Set-Occurrence und
– die Position innerhalb der Set-Occurrence.

Den einzufügenden Satz platziert UDS/SQL dann

– unmittelbar hinter dem CRS bei ORDER IS NEXT oder
– unmittelbar vor dem CRS bei ORDER IS PRIOR.

Wenn Sie sich für die automatische Auswahl der Set-Occurrence entscheiden (siehe Abschnitt „Auswahlmethode für Set-Occurrences bestimmen“ auf Seite 98), macht UDS/SQL den Ownersatz der Set-Occurrence automatisch zum CRS. In diesem Fall ist ORDER IS NEXT gleichbedeutend mit ORDER IS FIRST.

Auf Sets, die mit ORDER IS NEXT oder ORDER IS PRIOR definiert sind, können Sie mit SQL keinen INSERT und keinen UPDATE ausführen.

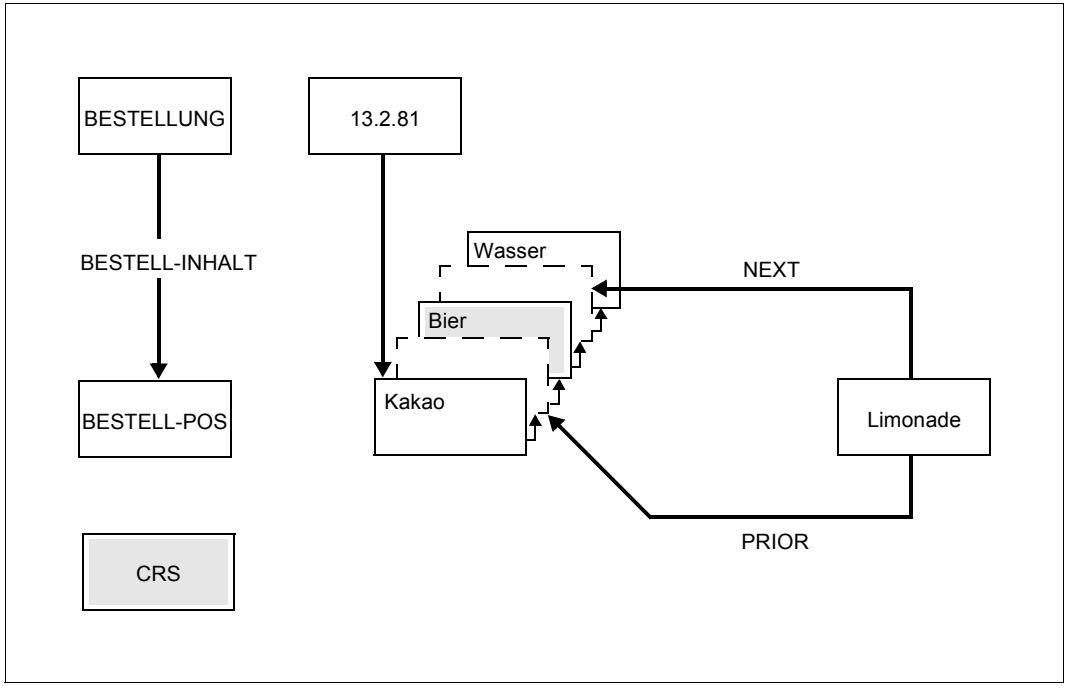


Bild 24: Satzfolge bei ORDER IS NEXT/PRIOR

ORDER IS IMMATERIAL

Sie überlassen UDS/SQL die Anordnung der Membersätze. Standardwerte für ORDER IS IMMATERIAL sind:

– bei der SSL-Angabe MODE IS CHAIN: ORDER IS NEXT
– bei der SSL-Angabe MODE IS LIST/POINTER ARRAY: ORDER IS LAST

Auf Sets, die mit ORDER IS IMMATERIAL definiert sind, können Sie mit SQL keinen INSERT und keinen UPDATE ausführen.

Sortierung der Membersätze nach den Werten des Primärschlüssels

```
                          ┌DATABASE-KEY                                    ┐
ORDER IS SORTED BY        │                                                │
      .                   └DEFINED KEYS DUPLICATES ARE[ NOT] ALLOWED       ┘
      .
      .
┌ASCENDING  ┐
│           │ KEY IS feldname,...
└DESCENDING ┘
```

ORDER IS SORTED BY DEFINED KEYS DUPLICATES ARE[NOT] ALLOWED

Felder, die Sie mit der ORDER-Klausel zu einem Schlüssel erklären, heißen Primärschlüssel des Set.
UDS/SQL sortiert die Membersätze der Set-Occurrences des Set aufsteigend (ASCENDING) oder absteigend (DESCENDING) nach den Werten des Primärschlüssels, der aus einem oder mehreren Feldern der Membersatzart besteht (besteht der Schlüssel aus mehreren Feldern, handelt es sich um einen Compound Key).

Mit *feldname* nennen Sie die Felder, aus denen Sie den Schlüssel zusammensetzen. Beim Einfügen neuer Membersätze in eine Set-Occurrence wählt UDS/SQL automatisch die Position aus, die dem Schlüsselwert entspricht. Wenn Sie Schlüsselwerte in der Datenbank ändern, aktualisiert UDS/SQL die Reihenfolge der Membersätze automatisch.

Mit DUPLICATES ARE[NOT] ALLOWED veranlassen Sie UDS/SQL, mehrfache Schlüsselwerte zurückzuweisen oder zuzulassen.

Bei DUPLICATES ARE ALLOWED erfolgt die Sortierung der Membersätze mit identischen Schlüsselwerten nach aufsteigenden Database-Key-Werten.

Der Primärschlüssel legt nicht nur die Reihenfolge der Membersätze innerhalb einer Set-Occurrence fest. Wenn Sie sich mit der SSL für eine Speicherungsart entscheiden, bei der UDS/SQL eine Tabelle anlegt, um die Sortierung einer Set-Occurrence herzustellen (siehe Seite 140, Adressliste, Liste), benutzt UDS/SQL die Tabelle auch als Direktzugriffspfad.

In diesem Fall müssen Sie die obige Klausel durch INDEXED ergänzen, damit die Tabelle mehrstufig angelegt wird (siehe Seite 95).

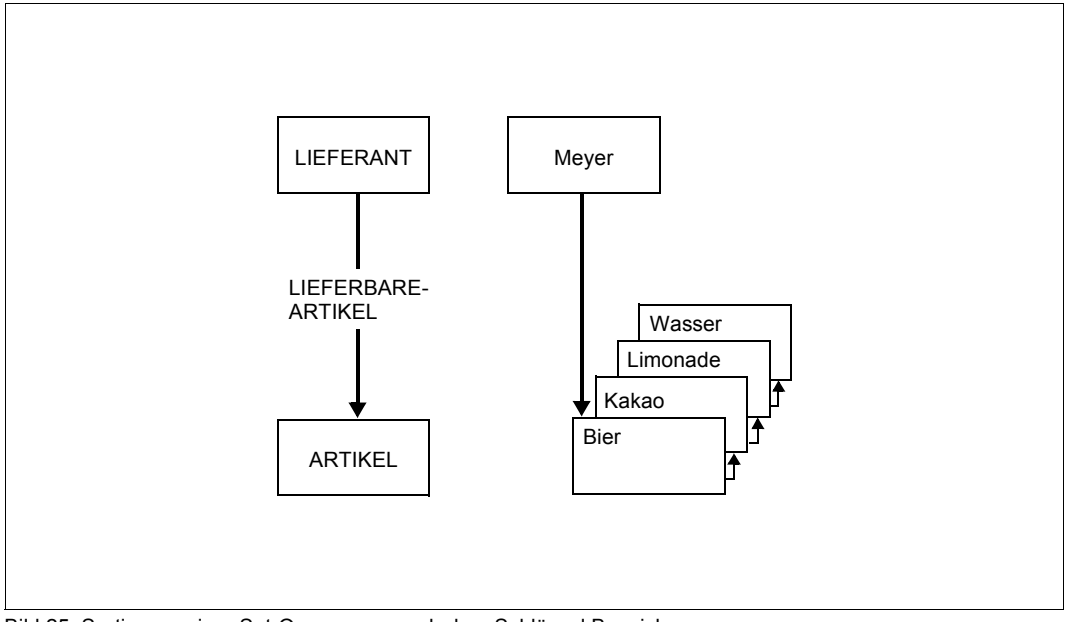


Bild 25: Sortierung einer Set-Occurrence nach dem Schlüssel Bezeichnung

ORDER IS SORTED BY DATABASE-KEY

UDS/SQL sortiert die Membersätze der Set-Occurrence des Set aufsteigend nach den Werten des Database Key. In diesem Fall wird der Database Key auch als Primärschlüssel des Set bezeichnet.

Sie können die Werte des Database Key selbst bestimmen (siehe Abschnitt „Vergabe der Database-Key-Werte durch den Anwender“ auf Seite 81) oder UDS/SQL die Zuordnung der Schlüsselwerte überlassen.

Wenn Sie sich mit der SSL für eine Speicherungsart entscheiden, bei der UDS/SQL eine Tabelle anlegt, um die Sortierung einer Set-Occurrence herzustellen (siehe Seite 140, Adressliste, Liste), benutzt UDS/SQL die Tabelle auch als Direktzugriffspfad.
In diesem Fall müssen Sie die obige Klausel durch INDEXED ergänzen, damit die Tabelle mehrstufig angelegt wird (siehe Seite 95).

4.7.4 Zusätzliche Zugriffspfade für Direktzugriff auf Setebene anlegen

Anders als auf Satzartebene unterstützt UDS/SQL den Direktzugriff auf Setebene nur über Tabellen. Lediglich die SYSTEM-Sets (siehe Abschnitt „SYSTEM-Set“ auf Seite 100) bieten Direktzugriff über ein Hashverfahren an. Sie haben zwei Möglichkeiten, Tabellen für den Direktzugriff auf Setebene zu definieren:

– Eine Tabelle, die den Primärschlüssel des Set enthält

– Eine oder mehrere Tabellen, die einen Sekundärschlüssel des Set (SEARCH-Key) enthalten.

Zusätzlichen Zugriffspfad über Primärschlüssel anlegen

```
ORDER IS SORTED INDEXED[ NAME IS name] BY

      {DATABASE-KEY                                }
      {DEFINED KEYS DUPLICATES ARE[ NOT] ALLOWED   }
        .
        .
        .
{ASCENDING }
{           } KEY IS feldname,...
{DESCENDING}
```

ORDER IS SORTED INDEXED BY DEFINED KEYS DUPLICATES ARE [NOT] ALLOWED ..... ASCENDING/DESCENDING KEY IS *feldname*,...

Mit dieser Angabe definieren Sie einerseits einen Primärschlüssel, der die Reihenfolge der Membersätze in den Set-Occurrences eines Set festlegt (siehe Abschnitt „Sortierung der Membersätze nach den Werten des Primärschlüssels“ auf Seite 93). Andererseits legen Sie für jede Set-Occurrence des Set eine mehrstufige Tabelle an. Die Tabelle stellt für jeden Satz, der zur Set-Occurrence gehört, eine eindeutige Beziehung zwischen Primärschlüsselwert, Database-Key-Wert und physischer Adresse des Satzes her. Die physische Adresse wird jedoch bei einer Lageänderung des Satzes nicht automatisch aktualisiert.

Mit *feldname* nennen Sie die Felder der Membersatzart, aus denen Sie den Schlüssel zusammensetzen.

Mit DUPLICATES ARE[NOT] ALLOWED veranlassen Sie UDS/SQL, mehrfache Schlüsselwerte zurückzuweisen oder zuzulassen.

Beispiel

```
RECORD NAME IS LIEFERANT
          .
          .
          .

RECORD NAME IS ARTIKEL
          .
          .
          .
01  BEZEICHNUNG PICTURE IS X(30).

SET NAME IS LIEFERBARE-ARTIKEL
    ORDER IS SORTED INDEXED BY DEFINED KEYS.....
    OWNER IS LIEFERANT.
MEMBER IS ARTIKEL.....
    ASCENDING KEY IS BEZEICHNUNG
              .
              .
              .
```

ORDER IS SORTED INDEXED BY DATABASE-KEY

Mit dieser Angabe definieren Sie einerseits eine Reihenfolge für die Membersätze der Set-Occurrences eines Set (siehe Abschnitt „Sortierung der Membersätze nach den Werten des Primärschlüssels“ auf Seite 93). Andererseits legen Sie für jede Set-Occurrence des Set eine mehrstufige Tabelle an. Die Tabelle stellt für jeden Satz, der zur Set-Occurrence gehört, eine eindeutige Beziehung zwischen Database-Key-Wert und physischer Adresse des Satzes her. Diesen Zugriffspfad benutzt UDS/SQL beim Ein- und Ausfügen eines Satzes innerhalb der Set-Occurrence und beim sequenziellen Lesen, nicht aber für Direktzugriffe.

Zusätzlichen Zugriffspfad über Sekundärschlüssel anlegen

```
SEARCH KEY IS feldname,... USING INDEX [NAME IS name]
            DUPLICATES ARE[ NOT] ALLOWED
```

Ein Schlüssel, den Sie mit SEARCH-KEY IS... erklären, heißt SEARCH-Key oder auch Sekundärschlüssel. Der Schlüssel kann aus mehreren Feldern zusammengesetzt sein.

Mit *feldname* nennen Sie die Felder, aus denen Sie den Schlüssel zusammensetzen, wobei alle genannten Felder Teil der Membersatzart sein müssen.

Mit *name* geben Sie den Namen der Tabelle an. In den die Tabelle betreffenden SSL-Anweisungen beziehen Sie sich auf diesen Namen.

Mit DUPLICATES ARE[NOT] ALLOWED entscheiden Sie, ob UDS/SQL gleiche Schlüsselwerte zurückweisen oder zulassen soll.

Auf Grund dieser Definition richtet UDS/SQL für jede Set-Occurrence des Set eine SEARCH-Key-Tabelle auf Setebene oder Set-SEARCH-Key-Tabelle ein. Diese Tabelle stellt für jeden Satz, der zur Set-Occurrence gehört, eine eindeutige Beziehung zwischen Sekundärschlüsselwert, Database-Key-Wert und physischer Adresse des Satzes her. Sie wird mehrstufig angelegt. Die physische Adresse wird bei einer Lageänderung des Satzes nicht automatisch aktualisiert.

Sie können unabhängig voneinander mehrere Sekundärschlüssel für einen Set definieren.

4.7.5 Auswahlmethode für Set-Occurrences bestimmen

```
SET OCCURRENCE SELECTION IS THRU

    {CURRENT OF SET                                                          }
    {                                                                        }
    {LOCATION MODE OF OWNER[ ALIAS FOR {feldname  } IS bezeichner-2]...      }
    {                                  {bezeichner-1}                        }
```

Der Zugriff auf Sätze über Sets erfordert die Auswahl der gewünschten Set-Occurrences.

Es stehen Ihnen zwei Auswahlmethoden zur Verfügung:

SET OCCURRENCE SELECTION IS THRU CURRENT OF SET

Hier identifiziert der zuletzt angesprochene Satz eines Set (CRS) die Set-Occurrence.

SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER
[ALIAS FOR *feldname* / *bezeichner-1* IS *bezeichner-2*]...

```
[ALIAS FOR {feldname  } IS bezeichner-2]...
           {bezeichner-1}
```

Diese Methode setzt voraus, dass Sie mit einer der folgenden Klauseln einen eindeutigen Primärschlüssel für die Ownersatzart definiert haben:

– LOCATION MODE IS DIRECT bzw. LOCATION MODE IS DIRECT-LONG

– LOCATION MODE IS CALC..... DUPLICATES ARE NOT ALLOWED

Dieser Schlüssel ist entweder der Database Key oder ein CALC-Key.

Diese Methode kann UDS/SQL verwenden, um einen Satz aus einer Set-Occurrence zu lesen (siehe Handbuch „Anwendungen programmieren“, FIND 7) oder um einen neuen Satz in eine Set-Occurrence einzuspeichern (siehe Handbuch „Anwendungen programmieren“, STORE, MODIFY). Der Programmierer hat nur die Aufgabe, an UDS/SQL den Schlüsselwert zu übergeben, der den Ownersatz der Set-Occurrence eindeutig bezeichnet. UDS/SQL wählt die Set-Occurrence dann automatisch mit einem Direktzugriff über den Database Key bzw. das Hashverfahren auf Ebene der Ownersatzart aus und macht den gefundenen Ownersatz zum CRS.

Automatisch mehrere Ownersätze aus derselben Satzart gleichzeitig auswählen

Bei einer bestimmten Datenstruktur kann der Fall eintreten, dass zum Einspeichern eines Membersatzes gleichzeitig mehrere Set-Occurrences, d.h. mehrere Ownersätze auszuwählen sind, wobei die Ownersätze alle derselben Satzart angehören.

Dieser Fall ist gegeben, wenn die beiden folgenden Bedingungen erfüllt sind:

– Zwei Satzarten sind durch mehr als einen Set miteinander verbunden.

– Die Membersatzart ist in mindestens zwei dieser Sets als AUTOMATIC-Member definiert.

UDS/SQL wählt die Set-Occurrences der Sets automatisch aus, wenn Sie die Auswahlmethode mit LOCATION MODE OF OWNER definiert haben. Für die gleichzeitige Auswahl von mehreren unterschiedlichen Ownersätzen benötigen Sie dann jedoch ein zusätzliches Sprachmittel, da es nicht möglich ist, die Felder, die Sie mit LOCATION MODE IS... zum Primärschlüssel definiert haben, gleichzeitig mit mehreren Schlüsselwerten zu versorgen. Mit der folgenden Klausel erzeugen Sie für die Ownerauswahl in diesem Set ein zusätzliches Feld, das die Schlüsselwerte aufnimmt:

```
ALIAS FOR { feldname      } IS bezeichner-2
          { bezeichner-1  }
```

---

feldname bzw. *bezeichner-1* bezeichnen Felder, die Sie mit LOCATION MODE IS... zum Primärschlüssel erklärt haben.

Mit *bezeichner-2* vergeben Sie den Namen für das zusätzlich zu erzeugende Feld. UDS/SQL richtet dieses Feld automatisch mit demselben Feldtyp und derselben Feldlänge ein wie das Feld *feldname* bzw. *bezeichner-1*.

Falls Sie mit LOCATION MODE IS CALC einen Schlüssel definiert haben, der aus mehreren Feldern zusammengesetzt ist, müssen Sie die ALIAS-Angabe so oft wiederholen, bis für jedes Schlüsselfeld genau ein Ersatzfeld vorhanden ist.

4.8 Spezielle Sets

4.8.1 SYSTEM-Set

```
OWNER IS SYSTEM.
```

Eine Satzart, die in Ihrer Datenstruktur keine Beziehung zu einer übergeordneten Satzart hat, können Sie trotzdem zum Member eines Set erklären.

Dies müssen Sie tun,

– wenn die Sätze zur sequenziellen Verarbeitung eine andere Reihenfolge einnehmen sollen, als aufsteigend nach Database-Key-Werten sortiert (siehe Seite 81 und Seite 90 ff).

– wenn Sie nur eine Auswahl der Sätze sequenziell verarbeiten wollen (siehe Abschnitt „Art der Set-Mitgliedschaft von Membersätzen definieren“ auf Seite 75).

In diesem Fall erklären Sie die Satzart zum Member eines Set, der nur eine symbolische Ownersatzart mit dem Namen SYSTEM besitzt. Solche Sets heißen deshalb SYSTEM-Sets.

Die symbolische Ownersatzart enthält genau einen Satz, der von UDS/SQL automatisch angelegt wird. Er wird als Ankersatz bezeichnet. Folglich besitzt ein SYSTEM-Set genau eine Set-Occurrence.

Zusätzlich zu den Möglichkeiten in einem normalen Set bieten SYSTEM-Sets den Direktzugriff über ein Hashverfahren.

Sekundärschlüssel zur Umrechnung durch Hashverfahren definieren

```
SEARCH KEY IS feldname,... USING CALC[ hashroutine]
    DUPLICATES ARE[ NOT] ALLOWED
```

Ein Schlüssel, den Sie mit SEARCH KEY IS... erklären, heißt SEARCH-Key oder auch Sekundärschlüssel. Der Schlüssel kann aus mehreren Feldern zusammengesetzt sein.

Mit *feldname* nennen Sie die Felder, aus denen Sie den Schlüssel zusammensetzen, wobei alle genannten Felder Teil der Membersatzart sein müssen.

Mit DUPLICATES ARE[NOT] ALLOWED entscheiden Sie, ob UDS/SQL gleiche Schlüsselwerte zurückweisen oder zulassen soll.

Mit *hashroutine* geben Sie den Namen eines Moduls an, der Ihren Sekundärschlüssel in eine vier byte lange Binärzahl umrechnet. Die Binärzahl setzt UDS/SQL anschließend in eine relative Seitennummer um. In dieser Seite findet UDS/SQL einen Zeiger auf den Satz (siehe Seite 211).

Wenn Sie zu *hashroutine* keine Angabe machen, verwendet UDS/SQL dieselbe Standard-Hashroutine wie zur Umrechnung des Primärschlüssels auf Satzartebene (zur Programmierung einer Hashroutine und zum Ablauf der Standard-Hashroutine siehe Seite 83 ff).

Sie können mehrere Sekundärschlüssel definieren.

4.8.2 Dynamischer Set

Die DML-Anweisungen operieren im Allgemeinen nur auf einem Satz der Datenbank. Es gibt jedoch eine DML-Anweisung zur Wiedergewinnung von Sätzen, die gleichzeitig mehrere Sätze aus der Datenbank auswählt (siehe Handbuch „Anwendungen programmieren“, FIND-7). Die ausgewählte Satzmenge wird zur weiteren Verarbeitung zwischengespeichert, indem UDS/SQL die Sätze automatisch zu Membersätzen in einem dynamischen Set macht.

Ein dynamischer Set ist also dadurch gekennzeichnet, dass er während einer Transaktion Sätze aus beliebigen Satzarten aufnehmen kann, um diese als Zwischenergebnis einer Suchfrage zu kennzeichnen, und wieder abgeben kann, wenn das Zwischenergebnis nicht mehr benötigt wird. Die Set-Mitgliedschaft im dynamischen Set ist daher vom Typ OPTIONAL MANUAL und es gibt keine bestimmte Membersatzart.

Zur Aufnahme ausgewählter Sätze in den dynamischen Set genügt eine Set-Occurrence, da es ja keine Verknüpfungsinformation zu bestimmten Ownersätzen zu speichern gibt, d.h. ein dynamischer Set wird als SYSTEM-Set erklärt.

Der Set heißt dynamisch, weil er nur während einer Transaktion Membersätze besitzt. Es existiert keine statische Set-Mitgliedschaft von Sätzen.

Wenn Sie einen dynamischen Set definieren, muss Ihr Schema einen Temporären Realm enthalten. Den dynamischen Set definieren Sie in der folgenden Form:

```
SET NAME IS setname
    SET IS DYNAMIC
    ORDER IS IMMATERIAL
    OWNER IS SYSTEM.
```

Wenn Sie die Dialogsprache IQL einsetzen wollen, müssen Sie acht dynamische Sets definieren, denen Sie die Namen IQL-DYN1 bis IQL-DYN8 geben müssen. Näheres finden Sie im Handbuch „Dialogsystem IQS“.

4.9 Hashbereiche und Tabellen benennen

Hashbereichen und Tabellen für Sekundärschlüssel sowie Tabellen für Primärschlüssel müssen Sie einen Namen geben, wenn Sie sie in der SSL zu folgenden Zwecken ansprechen wollen:

– Physische Lage der Hashbereiche und Tabellen bestimmen
– Redundanz in den Tabellen verhindern
– Reorganisationsaufwand der Tabellen festlegen

Sie benennen Hashbereiche oder Tabellen für Sekundärschlüssel auf Satzart- und auf Setebene durch:

SEARCH KEY IS..... USING { CALC / INDEX } NAME IS *name*.....

Sie benennen Tabellen für Primärschlüssel auf Setebene durch:

ORDER IS SORTED INDEXED NAME IS *name*.....

Mit *name* vergeben Sie den Namen für den Hashbereich bzw. die Tabelle.

4.10 Das Realm-Konzept

Um Aspekte

- des Datenschutzes
- der Datensicherung
- des konkurrierenden Zugriffs auf die Daten und
- der logischen Zusammengehörigkeit bestimmter Daten

berücksichtigen zu können, ist es nützlich, die Datenbank in Untereinheiten aufzuteilen. Die Untereinheiten heißen Realms (auch Areas genannt). Sie werden als BS2000-Dateien beim Aufbau der Datenbank erzeugt (siehe Handbuch „Aufbauen und Umstrukturieren“, Datenbank aufbauen). Es gibt Realms, die ausschließlich interne Informationen von UDS/SQL enthalten, und solche, in denen Ihre Daten gespeichert sind. Die letztgenannten heißen Benutzerrealms.

Bei einer Datenbank mit 2048 byte Seitenlänge (2-Kbyte-Seitenformat) können Sie maximal 123 Realms definieren.
Bei einer Datenbank mit 4000 byte oder 8096 byte Seitenlänge (4-Kbyte- bzw. 8-Kbyte-Seitenformat) können Sie maximal 245 Realms definieren.

Datenschutz

Mit dem Dienstprogramm BPRIVACY kann der Datenbankadministrator die Zugriffsrechte bestimmter Benutzergruppen für DB-Objekte (Realms, Satzarten, Sets) vergeben.
Bei der Definition eines Subschemas können Sie die Benutzung von Daten Realm-weise erlauben oder verbieten.

Datensicherung

Durch die Untergliederung der Datenbank in Realms ist es möglich, die Auswirkungen von Gerätefehlern oder Schreib-Lesefehlern auf wenige Realms oder nur einen Realm einzugrenzen. Es genügt dann, nur die betroffenen Realms bzw. den betroffenen Realm mit Hilfe von Realm-Sicherungen und evtl. vorhandenen After-Images zu rekonstruieren (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BMEND). Daher empfiehlt es sich, Daten, die häufig zu ändern sind, in einen anderen Realm zu speichern als Daten, die selten oder nur zu bestimmten Zeiten verändert werden.

Steuerung des konkurrierenden Zugriffs

Bei der Eröffnung einer Transaktion gibt der Datenbankprogrammierer an, auf welche Realms die Transaktion zugreifen muss (siehe Handbuch „Anwendungen programmieren“, READY). Er kann dabei Benutzungsarten für die Realms definieren, die die gleichzeitigen Zugriffe anderer Transaktionen auf diese Realms einschränken oder ganz ausschließen. Durch Aufteilen des Datenbestandes auf mehrere Realms wird die Behinderung anderer Transaktionen so gering wie möglich gehalten.

Wenn Sie Daten, die oft gleichzeitig benötigt werden, in verschiedene Realms speichern und die Realms verschiedenen Plattenlaufwerken zuordnen, können sich die Datenzugriffe verschiedener Transaktionen zeitlich überlappen. Sie erzielen kürzere Zugriffszeiten.

Logische Zusammengehörigkeit von Daten

Sie können Ihren Datenbestand so auf verschiedene Realms aufteilen, dass Ihre Programme jeweils nur einen Teil der Realms verarbeiten. Je nach Bedarf können Sie die Realms mit dem Dienstprogramm BMEND bzw. in einer Session über die Administration mit DAL-Kommandos aus- oder anschließen. Selten benötigte Realms brauchen dann nicht ständig Ihre Betriebsmittel beanspruchen.

4.10.1 Definieren eines Realms

```
AREA NAME IS realmname
```

Mit *realmname* benennen Sie den Realm. Weitere Angaben sind zur Definition eines Realm nicht nötig. Insbesondere ist die Größe des Realm erst beim Aufbau der Datenbank anzugeben (siehe Handbuch „Aufbauen und Umstrukturieren“, Datenbank aufbauen).

4.10.2 Verteilung von Sätzen auf Realms bestimmen

```
RECORD NAME IS satzname
    WITHIN realmname-1[,realmname-2,... AREA-ID IS bezeichner]
```

Die Verteilung von Daten auf Realms und die Platzierung der Daten innerhalb von Realms nehmen Sie im Wesentlichen bei der Beschreibung der physischen Speicherstruktur mit der SSL vor (siehe Abschnitt „Lage von Membersätzen, Tabellen und Hashbereichen bestimmen“ auf Seite 157). Lediglich die Verteilung von Sätzen auf Realms definieren Sie mit der Schema-DDL.

Mit *realmname-1* usw. zählen Sie alle Realms auf, die Sätze der Satzart *satzname* aufnehmen sollen. Wenn Sie mehrere Realms angeben, können Sie mit SQL keinen neuen Satz dieser Satzart einfügen.
Wenn die Satzart *satzname* Membersatzart einer verteilbaren Liste ist (siehe Seite 145), zählen Sie mit *realmname-1* usw. alle Realms auf, in denen die Sätze abgespeichert werden können. Sofern die Lage des Tabellenteils (Seiten mit Stufe > 0) der verteilbaren Liste nicht explizit bestimmt ist (MODE IS LIST DETACHED WITHIN ...), bestimmt *realmname-1* die Lage des Tabellenteils implizit.

bezeichner brauchen Sie nur, wenn Sie mehr als einen Realm-Namen nennen.
Mit *bezeichner* vergeben Sie den Namen für ein Feld, das UDS/SQL automatisch erzeugt, um jeweils einen der aufgezählten Realm-Namen aufzunehmen. Der Datenbankprogrammierer muss vor dem Einspeichern eines Satzes in das Feld den Namen des Realm übertragen, der den Satz aufnehmen soll (siehe Handbuch „Anwendungen programmieren“, STORE). Wenn der Satz Membersatzart einer verteilbaren Liste ist, müssen Sie keinen Realm-Namen im Feld *bezeichner* versorgen. Der Inhalt des Feldes wird beim Einspeichern ignoriert.

Die Wiedergewinnung von Membersätzen über ihre Set-Beziehung erfordert weniger Zeit, wenn Sie die Membersätze in denselben Realm wie den zugehörigen Ownersatz speichern. In diesem Fall können Sie mit der SSL eine noch weiter optimierte Platzierung der Sätze vereinbaren (siehe Abschnitt „Lagebestimmung innerhalb eines Realm“ auf Seite 160).

4.10.3 Temporärer Realm

```
AREA IS TEMPORARY
```

Einen Temporären Realm müssen Sie immer definieren, wenn Ihre Schemabeschreibung dynamische Sets enthält, oder wenn UDS/SQL auf Grund einer DML-Anweisung (siehe Handbuch „Anwendungen programmieren“, FIND 7) automatisch einen dynamischen Set anlegen muss. Der Temporäre Realm dient zur Aufnahme der Tabelle, die die Set-Occurrence des dynamischen Sets repräsentiert und auf die zugehörigen Membersätze verweist. Einen Temporären Realm müssen Sie auch definieren, wenn Sie mit SQL auf die Datenbank zugreifen wollen.

4.11 Benennung und Schutz des Schemas

```
SCHEMA NAME IS schemaname
    [PRIVACY LOCK FOR COPY IS literal-1[ OR literal-2]].
```

Die Beschreibung einer Datenbank mit der Schema-DDL beginnt immer mit der Benennung des Schemas.

Mit *schemaname* geben Sie dem Schema einen Namen. Auf diesen Namen nehmen Sie später mit der SSL, der Subschema-DDL und den Dienstprogrammen Bezug.

Die Benutzung eines Subschemas zum Zugriff auf die Daten ist abhängig von den Zugriffsrechten des jeweiligen Benutzers.
Mit *literal-1* und *literal-2* vergeben Sie Kennwörter, die das unbefugte Erstellen eines Subschemas verhindern. Ein Subschema kann dann erzeugt werden, wenn mindestens eines der Kennwörter bekannt ist.

4.12 Gesamtbeispiel für DDL

Das Beispiel zeigt das Schema eines Artikelversands. Auf diesem Schema können folgende Funktionen ablaufen:

- Stammdatenverwaltung über Kunden, Artikel, Lieferanten, von Kunden erteilte Aufträge und bei Lieferanten bestellte Artikel
- Lagerbestandsführung
- Mahnwesen
- Stücklistenverarbeitung für Ersatzteile

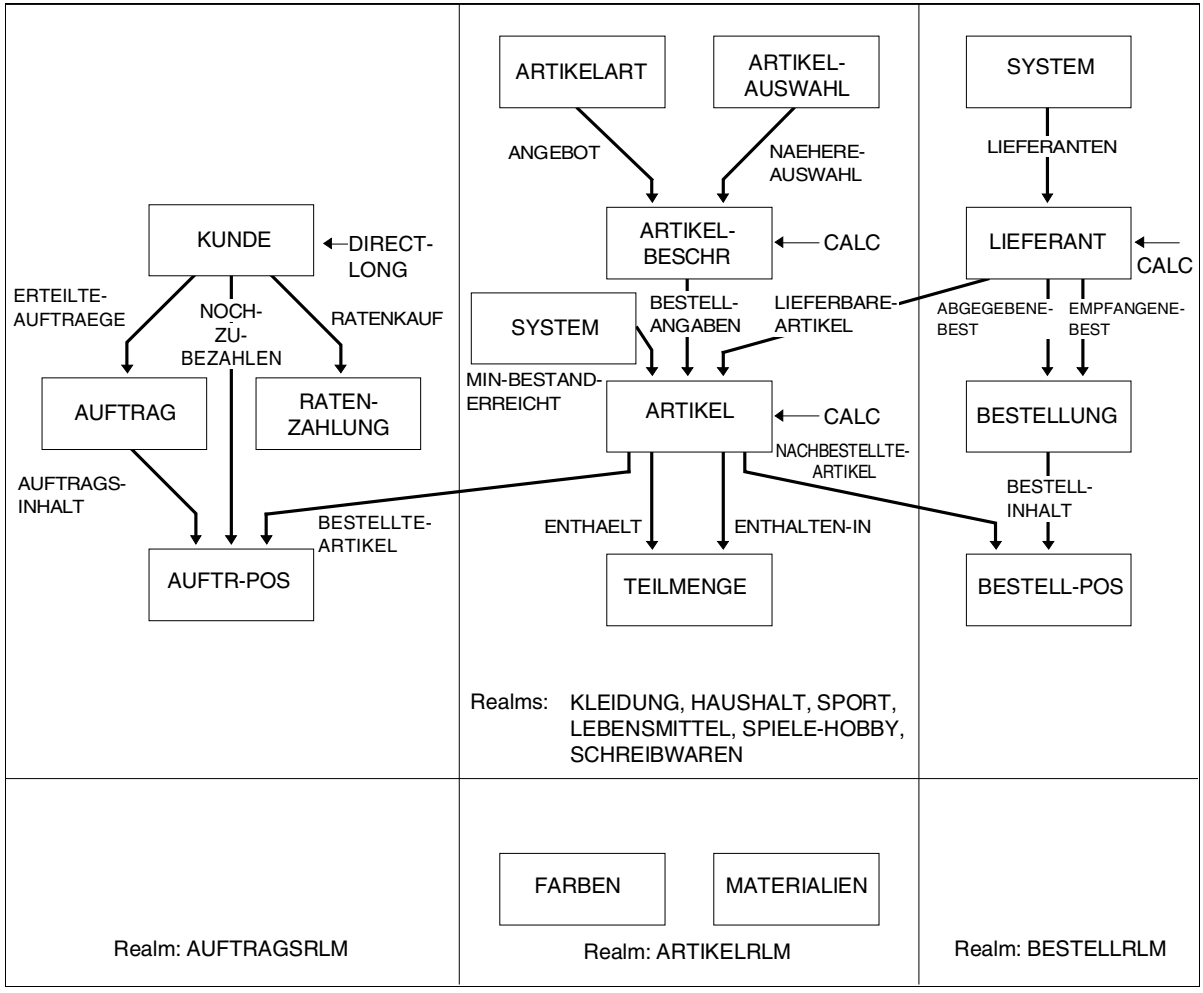


Bild 26: Schema eines Artikelversands

Eine Grobauswahl von Artikeln ist über die Auswahlkriterien ARTIKELART und ARTIKELAUSWAHL möglich. Diese Auswahl führt zu einer detaillierten Artikelbeschreibung. Zu einer Artikelbeschreibung können mehrere Artikel gehören, die sich in Farbe, Größe und Preis unterscheiden können.
Mit den Sets ENTHAELT und ENTHALTEN IN ist eine Stückliste für die Ersatzteilversorgung realisiert.
In den Artikelsätzen sind Farben durch eine Nummer, Materialien durch Kürzel mit Prozentangaben verschlüsselt. In den Satzarten FARBE und MATERIALIEN sind die Zuordnungen des Codes zur zugehörigen Klartextangabe gespeichert.

Erweiterungen des Schemas wurden vorgenommen, um möglichst vollständig die Anwendung der DDL-Klauseln zu zeigen.

```
 1              SCHEMA NAME IS                        ARTIKELVERSAND
 2                  PRIVACY LOCK FOR COPY IS          "VERSANDKEY".
 3             *
 4             *
 5             *
 6              AREA NAME IS                          AUFTRAGSRLM.
 7              AREA NAME IS                          BESTELLRLM.
 8              AREA NAME IS                          KLEIDUNG.
 9              AREA NAME IS                          HAUSHALT.
10              AREA NAME IS                          SPORT.
11              AREA NAME IS                          LEBENSMITTEL.
12              AREA NAME IS                          SPIELE-HOBBY.
13              AREA NAME IS                          SCHREIBWAREN.
14              AREA NAME IS                          ARTIKELRLM.
15              AREA NAME IS                          SUCHRLM
16                                                    AREA IS TEMPORARY.
17             *
18             *
19             *
20              RECORD NAME IS                        KUNDE
21                  LOCATION MODE IS DIRECT-LONG KUNDEN-NR OF KUNDE
22                  WITHIN AUFTRAGSRLM.
23             *
24              01  KUNDEN-NAME                      TYPE IS CHARACTER 30.
25              01  KUNDEN-VORNAME                   TYPE IS CHARACTER 30.
26              01  KUNDEN-NR                        TYPE IS DATABASE-KEY-LONG.
27             *
28             *
29              RECORD NAME IS                        AUFTRAG
30                  WITHIN AUFTRAGSRLM.
31             *
32              01  AUFTR-NR                         PICTURE IS 9(4).
33              01  AUFTR-JAHR                       PICTURE IS 99.
34              01  AUFTR-MONAT                      PICTURE IS 99.
35              01  AUFTR-TAG                        PICTURE IS 99.
36              01  AUFTR-STATUS                     PICTURE IS X.
37             *
```

```
38              *
39               RECORD NAME IS                        AUFTR-POS
40                   WITHIN AUFTRAGSRLM.
41              *
42               01  AUFTR-POS-NR                     PICTURE IS 99.
43               01  AUFTR-MENGE                      TYPE IS DECIMAL 6.
44               01  KENNZ-RATENZAHLUNG               PICTURE IS X.
45               01  AUFTR-POS-STATUS                 PICTURE IS X.
46              *
47              *
48               RECORD NAME IS                        RATENZAHLUNG
49                   WITHIN AUFTRAGSRLM
50                   SEARCH KEY IS NEXT-RATE-JAHR, NEXT-RATE-MONAT, NEXT-RATE-TAG
51                      USING INDEX NAME IS SEARCH-TAB-RATENZAHLUNG
52                      DUPLICATES ARE ALLOWED.
53              *
54               01  AUFTR-NR                         PICTURE IS 9(4).
55               01  AUFTR-POS-NR                     PICTURE IS 99.
56               01  GESAMTPREIS-RATE                 TYPE IS DECIMAL 9,2.
57               01  EINZELRATE                       TYPE IS DECIMAL 7,2.
58               01  RESTBETRAG                       TYPE IS DECIMAL 9,2.
59               01  NEXT-RATE-JAHR                   PICTURE IS 99.
60               01  NEXT-RATE-MONAT                  PICTURE IS 99.
61               01  NEXT-RATE-TAG                    PICTURE IS 99.
62              *
63              *
64               RECORD NAME IS                        ARTIKELART
65                   WITHIN KLEIDUNG, HAUSHALT, SPORT, LEBENSMITTEL, SPIELE-HOBBY,
66                      SCHREIBWAREN AREA-ID IS RLMAUSWAHL-1
67                   SEARCH KEY IS ART-BEZ USING CALC
68                      NAME IS SEARCH-TAB-ARTIKELART DUPLICATES ARE ALLOWED.
69              *
70               01  ART-BEZ                          TYPE IS CHARACTER 25.
71              *
72              *
73               RECORD NAME IS                        ARTIKELAUSWAHL
74                   WITHIN KLEIDUNG, HAUSHALT, SPORT, LEBENSMITTEL, SPIELE-HOBBY,
75                      SCHREIBWAREN AREA-ID IS RLMAUSWAHL-2
76                   SEARCH KEY IS WAHL-KRIT USING INDEX
77                      NAME IS SEARCH-TAB-ARTIKELAUSWAHL DUPLICATES ARE ALLOWED.
78              *
79               01  WAHL-KRIT                        TYPE IS CHARACTER 25.
80              *
81              *
82               RECORD NAME IS                        ARTIKELBESCHR
83                   LOCATION MODE IS CALC USING BEZEICHNUNG
84                      DUPLICATES ARE ALLOWED
85                   WITHIN KLEIDUNG, HAUSHALT, SPORT, LEBENSMITTEL, SPIELE-HOBBY,
86                      SCHREIBWAREN AREA-ID IS RLMAUSWAHL-3.
87              *
88               01  ART-NR                           PICTURE IS 9(6).
89               01  BEZEICHNUNG                      TYPE IS CHARACTER 40.
```

```
 90              01  MATERIAL                        OCCURS 4 TIMES.
 91                  02 PROZENT                      PICTURE IS 99.
 92                  02 MAT-ABK                      PICTURE IS X.
 93              01  LAENGENFELD                     TYPE IS BINARY 15.
 94              01  ARTIKELERLAEUTERUNG             PICTURE IS LX(500)
 95                                                  DEPENDING ON LAENGENFELD.
 96             *
 97             *
 98              RECORD NAME IS                      ARTIKEL
 99                  LOCATION MODE IS CALC USING ART-NR, FARB-NR, GROESSE
100                     DUPLICATES ARE NOT ALLOWED
101                  WITHIN KLEIDUNG, HAUSHALT, SPORT, LEBENSMITTEL, SPIELE-HOBBY,
102                     SCHREIBWAREN AREA-ID IS RLMAUSWAHL-4
103                  SEARCH KEY IS ART-NR-LIEFER, FARB-NR-LIEFER, GROESSE
104                     USING CALC NAME IS SEARCH-TAB-ARTIKEL-1
105                     DUPLICATES ARE NOT ALLOWED
106                  SEARCH KEY IS BEZEICHNUNG USING CALC
107                     NAME IS SEARCH-TAB-ARTIKEL-2 DUPLICATES ARE ALLOWED.
108             *
109              01  ART-NR                          PICTURE IS 9(6).
110              01  FARB-NR                         PICTURE IS 99.
111              01  BEZEICHNUNG                     TYPE IS CHARACTER 40.
112              01  ART-NR-LIEFER                   PICTURE IS 9(4).
113              01  FARB-NR-LIEFER                  PICTURE IS 99.
114              01  GROESSE                         PICTURE IS 99.
115              01  PREIS                           TYPE IS DECIMAL 7,2.
116              01  PREIS-RATENZAHLUNG              TYPE IS DECIMAL 7,2.
117              01  MAX-BESTAND                     TYPE IS DECIMAL 10.
118              01  MIN-BESTAND                     TYPE IS DECIMAL 3.
119              01  AKT-BESTAND                     TYPE IS DECIMAL 10.
120              01  STATISTIK                       TYPE IS DECIMAL 15.
121              01  KENNZ-NICHT-LIEFERBAR           PICTURE IS X.
122             *
123             *
124              RECORD NAME IS                      TEILMENGE
125                  WITHIN HAUSHALT, SPORT AREA-ID IS RLMAUSWAHL-5.
126             *
127              01  MENGE                           PICTURE IS 99.
128             *
129             *
130              RECORD NAME IS                      FARBEN
131                  WITHIN ARTIKELRLM
132                  SEARCH KEY IS FARB-BEZ USING CALC DUPLICATES ARE NOT ALLOWED
133                  SEARCH KEY IS FARB-NR USING CALC DUPLICATES ARE NOT ALLOWED.
134             *
135              01  FARB-NR                         PICTURE IS 99.
136              01  FARB-BEZ                        TYPE IS CHARACTER 20.
137             *
138             *
139              RECORD NAME IS                      MATERIALIEN
140                  WITHIN ARTIKELRLM
141                  SEARCH KEY IS MAT-ABK USING INDEX
```

```
142                    NAME IS SEARCH-TAB-MATERIAL-1 DUPLICATES ARE NOT ALLOWED
143                SEARCH KEY IS MAT-BEZ USING INDEX
144                    NAME IS SEARCH-TAB-MATERIAL-2 DUPLICATES ARE NOT ALLOWED.
145            *
146             01  MAT-ABK                          TYPE IS CHARACTER 1.
147             01  MAT-BEZ                          TYPE IS CHARACTER 20.
148            *
149            *
150             RECORD NAME IS                       LIEFERANT
151                 LOCATION MODE IS CALC USING LIEFER-NR, LIEFER-NAME
152                    DUPLICATES ARE NOT ALLOWED
153                 WITHIN BESTELLRLM.
154            *
155             01  LIEFER-NR                        PICTURE IS 9(5).
156             01  LIEFER-NAME                      TYPE IS CHARACTER 30.
157             01  LIEFER-PLZ                       TYPE IS CHARACTER 4.
158             01  LIEFER-STADT                     TYPE IS CHARACTER 30.
159             01  LIEFER-STRASSE                   TYPE IS CHARACTER 30.
160             01  LIEFER-HAUSNR                    TYPE IS CHARACTER 3.
161             01  LIEFER-TEL                       PICTURE IS 9(12).
162             01  LIEFER-POSTFACH                  PIC 9(4).
163             01  LIEFER-FERNSCHR                  PIC 9(12).
164            *
165            *
166             RECORD NAME IS                       BESTELLUNG
167                 WITHIN BESTELLRLM.
168            *
169             01  BEST-NR                          PICTURE IS 9(4).
170             01  BEST-JAHR                        PICTURE IS 99.
171             01  BEST-MONAT                       PICTURE IS 99.
172             01  BEST-TAG                         PICTURE IS 99.
173            *
174            *
175             RECORD NAME IS                       BESTELL-POS
176                 WITHIN BESTELLRLM.
177            *
178             01  BEST-POS-NR                      PICTURE IS 99.
179             01  BEST-MENGE                       TYPE IS DECIMAL 10.
180            *
181            *
182            *
183             SET NAME IS                          ERTEILTE-AUFTRAEGE
184                 ORDER IS SORTED INDEXED BY DEFINED KEYS
185                    DUPLICATES ARE NOT ALLOWED
186                 OWNER IS KUNDE.
187             MEMBER IS AUFTRAG OPTIONAL AUTOMATIC
188                 ASCENDING KEY IS AUFTR-NR
189                 SEARCH KEY IS AUFTR-JAHR, AUFTR-MONAT, AUFTR-TAG USING INDEX
190                    NAME IS SEARCH-TAB-ERT-AUFTR DUPLICATES ARE ALLOWED
191                 SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
192            *
193            *
```

```
194              SET NAME IS                          AUFTR-INHALT
195                  ORDER IS SORTED INDEXED BY DEFINED KEYS
196                     DUPLICATES ARE NOT ALLOWED
197                  OWNER IS AUFTRAG.
198              MEMBER IS AUFTR-POS MANDATORY AUTOMATIC
199                  ASCENDING KEY IS AUFTR-POS-NR
200                  SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
201             *
202             *
203              SET NAME IS                          NOCH-ZU-BEZAHLEN
204                  ORDER IS LAST
205                  OWNER IS KUNDE.
206              MEMBER IS AUFTR-POS OPTIONAL AUTOMATIC
207                  SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
208             *
209             *
210              SET NAME IS                          RATENKAUF
211                  ORDER IS LAST
212                  OWNER IS KUNDE.
213              MEMBER IS RATENZAHLUNG MANDATORY AUTOMATIC
214                  SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
215             *
216             *
217              SET NAME IS                          ANGEBOT
218                  ORDER IS SORTED INDEXED BY DEFINED KEYS
219                     DUPLICATES ARE ALLOWED
220                  OWNER IS ARTIKELART.
221              MEMBER IS ARTIKELBESCHR MANDATORY AUTOMATIC
222                  ASCENDING KEY IS BEZEICHNUNG
223                  SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
224             *
225             *
226              SET NAME IS                          NAEHERE-AUSWAHL
227                  ORDER IS SORTED INDEXED BY DEFINED KEYS
228                     DUPLICATES ARE ALLOWED
229                  OWNER IS ARTIKELAUSWAHL.
230              MEMBER IS ARTIKELBESCHR MANDATORY AUTOMATIC
231                  ASCENDING KEY IS BEZEICHNUNG
232                  SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
233             *
234             *
235              SET NAME IS                          BESTELLANGABEN
236                  ORDER IS SORTED INDEXED BY DEFINED KEYS
237                     DUPLICATES ARE NOT ALLOWED
238                  OWNER IS ARTIKELBESCHR.
239              MEMBER IS ARTIKEL MANDATORY AUTOMATIC
240                  ASCENDING KEY IS FARB-NR, GROESSE
241                  SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
242             *
243             *
244              SET NAME IS                          MIN-BESTAND-ERREICHT
245                  ORDER IS SORTED INDEXED BY DEFINED KEYS
```

```
246                DUPLICATES ARE NOT ALLOWED
247             OWNER IS SYSTEM.
248          MEMBER IS ARTIKEL OPTIONAL MANUAL
249             ASCENDING KEY IS ART-NR, FARB-NR, GROESSE.
250         *
251         *
252          SET NAME IS                          ENTHAELT
253             ORDER IS NEXT
254             OWNER IS ARTIKEL.
255          MEMBER IS TEILMENGE MANDATORY AUTOMATIC
256             SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
257         *
258         *
259          SET NAME IS                          ENTHALTEN-IN
260             ORDER IS NEXT
261             OWNER IS ARTIKEL.
262          MEMBER IS TEILMENGE MANDATORY AUTOMATIC
263             SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER
264                ALIAS FOR   ART-NR     IS   ERSATZ-ART-NR
265                ALIAS FOR   FARB-NR    IS   ERSATZ-FARB-NR
266                ALIAS FOR   GROESSE    IS   ERSATZ-GROESSE.
267         *
268         *
269          SET NAME IS                          LIEFERANTEN
270             ORDER IS SORTED INDEXED BY DEFINED KEYS
271                DUPLICATES ARE NOT ALLOWED
272             OWNER IS SYSTEM.
273          MEMBER IS LIEFERANT MANDATORY AUTOMATIC
274             ASCENDING KEY IS LIEFER-NAME, LIEFER-NR.
275         *
276         *
277          SET NAME IS                          LIEFERBARE-ARTIKEL
278             ORDER IS SORTED INDEXED BY DEFINED KEYS
279                DUPLICATES ARE ALLOWED
280             OWNER IS LIEFERANT.
281          MEMBER IS ARTIKEL MANDATORY AUTOMATIC
282             ASCENDING KEY IS BEZEICHNUNG
283             SEARCH KEY IS KENNZ-NICHT-LIEFERBAR USING INDEX
284                NAME IS SEARCH-TAB-LIEF-ART DUPLICATES ARE ALLOWED
285             SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
286         *
287         *
288          SET NAME IS                          BESTELLTE-ARTIKEL
289             ORDER IS LAST
290             OWNER IS ARTIKEL.
291          MEMBER IS AUFTR-POS MANDATORY AUTOMATIC
292             SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
293         *
294         *
295          SET NAME IS                          NACHBESTELLTE-ARTIKEL
296             ORDER IS LAST
297             OWNER IS ARTIKEL.
```

```
298              MEMBER IS BESTELL-POS MANDATORY AUTOMATIC
299                  SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
300             *
301             *
302              SET NAME IS                          ABGEGEBENE-BEST
303                  ORDER IS LAST
304                  OWNER IS LIEFERANT.
305              MEMBER IS BESTELLUNG MANDATORY AUTOMATIC
306                  SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
307             *
308             *
309              SET NAME IS                          EMPFANGENE-BEST
310                  ORDER IS FIRST
311                  OWNER IS LIEFERANT.
312              MEMBER IS BESTELLUNG MANDATORY MANUAL
313                  SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
314             *
315             *
316              SET NAME IS                          BESTELL-INHALT
317                  ORDER IS NEXT
318                  OWNER IS BESTELLUNG.
319              MEMBER IS BESTELL-POS MANDATORY AUTOMATIC
320                  SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
321             *
322             *
323             *
324              SET NAME IS                          ERGEBNISSET
325                  SET IS DYNAMIC
326                  ORDER IS IMMATERIAL
327                  OWNER IS SYSTEM.
328             *
329              SET NAME IS                          LIMITED-SET
330                  SET IS DYNAMIC
331                  ORDER IS IMMATERIAL
332                  OWNER IS SYSTEM.
333             *
334             *
335              SET NAME IS                          IQL-DYN1
336                  SET IS DYNAMIC
337                  ORDER IS IMMATERIAL
338                  OWNER IS SYSTEM.
339             *
340              SET NAME IS                          IQL-DYN2
341                  SET IS DYNAMIC
342                  ORDER IS IMMATERIAL
343                  OWNER IS SYSTEM.
344             *
345              SET NAME IS                          IQL-DYN3
346                  SET IS DYNAMIC
347                  ORDER IS IMMATERIAL
348                  OWNER IS SYSTEM.
349             *
```

```
350              SET NAME IS                          IQL-DYN4
351                  SET IS DYNAMIC
352                  ORDER IS IMMATERIAL
353                  OWNER IS SYSTEM.
354             *
355              SET NAME IS                          IQL-DYN5
356                  SET IS DYNAMIC
357                  ORDER IS IMMATERIAL
358                  OWNER IS SYSTEM.
359             *
360              SET NAME IS                          IQL-DYN6
361                  SET IS DYNAMIC
362                  ORDER IS IMMATERIAL
363                  OWNER IS SYSTEM.
364             *
365              SET NAME IS                          IQL-DYN7
366                  SET IS DYNAMIC
367                  ORDER IS IMMATERIAL
368                  OWNER IS SYSTEM.
369             *
370              SET NAME IS                          IQL-DYN8
371                  SET IS DYNAMIC
372                  ORDER IS IMMATERIAL
373                  OWNER IS SYSTEM.
```

4.13 Reservierte Wörter des DDL-Compilers

ACCEPT	ACCESS	ACTUAL
AD	ADD	ADVANCING
AFTER	ALIAS	ALL
ALLOWED	ALPHABETIC	ALPHANUMERIC
ALSO	ALTER	ALTERNATE
AN	AND	ANY
APPLY	ARE	AREA
AREA-ID	AREAS	ASC
ASCENDING	ASSIGN	ASSIGNED
AT	ATTACHED	AUTHOR
AUTO	AUTOMATIC	BEFORE
BEGINNING	BETWEEN	BIN
BINARY	BLANK	BLOCK
BLOCK-DENSITY	BOTTOM	BY
C01	C02	C03
C04	C05	C06
C07	C08	C09
C10	C11	C12
CALC	CALL	CANCEL
CARD-PUNCH	CARD-READER	CBL-CTR
CD	CH	CHAIN
CHANGED	CHAR	CHARACTER
CHARACTERS	CHECK	CHECKING
CHECKPOINT	CLASS	CLOCK-UNITS
CLOSE	COBOL	CODE
CODE-SET	COLLATING	COLUMN
COMMA	COMMUNICATION	COMP
COMP-1	COMP-2	COMP-3
COMPILE	COMPRESSION	COMPUTATIONAL
COMPUTATIONAL-1	COMPUTATIONAL-2	COMPUTATIONAL-3

COMPUTE	CONFIGURATION	CONNECT
CONSOLE	CONTAINS	CONTROL
CONTROLS	COPY	CORR
CORRESPONDING	COUNT	CREATING
CSP	CURRENCY	CURRENT
CURRENT-DATE	CYCLES	CYLINDER-OFLO
DATA	DATABASE-EXCEPTION	DATABASE-KEY
DATABASE-KEY-LIST	DATABASE-KEY-LONG	DATABASE-KEY-NAME
DATABASE-KEY-RANGE	DATABASE-KEY-TRANSLATION-TABLE	DATABASE-PRIVACY-KEY
DATABASE-REALM-NAME	DATABASE-RECORD-NAME	DATABASE-SET-NAME
DATABASE-STATUS	DATE	DATE-COMPILED
DATE-WRITTEN	DAY	DB
DBKEY	DBKEY-LONG	DBKEY-TRANSLATION-TABLE
DBTT	DCB-NAME	DE
DEBUGGING	DEC	DECIMAL
DECIMAL-POINT	DECLARATIVES	DEFINED
DELETE	DELIMETED	DELIMETER
DEPENDING	DESC	DESCENDING
DESTINATION	DETACHED	DETAIL
DIGITS	DIRECT	DIRECT-LONG
DISABLE	DISC	DISC64
DISC80	DISC90	DISCONNECT
DISPLAY	DISPLAY-ST	DIVIDE
DIVISION	DOWN	DUP
DUPLICATE	DUPLICATES	DYNAMIC
EGI	ELSE	EMI
EMPTY	ENABLE	END
END-OF-PAGE	ENDING	ENTER
ENTRY	ENVIRONMENT	EOP
EQUAL	ERASE	ERROR
ESI	EVERY	EXAMINE

EXCEPTION	EXCL	EXCLUSIVE
EXHIBIT	EXIT	EXTEND
EXTENDED	FD	FETCH
FILE	FILE-CONTROL	FILE-LIMIT
FILE-LIMITS	FILES	FILLER
FINAL	FIND	FINISH
FIRST	FIXED	FOOTING
FOR	FORM-OVERFLOW	FREE
FROM	GENERATE	GET
GIVING	GO	GREATER
GREATEST	GROUP	GROUP-USAGE
HEADING	HIGH-VALUE	HIGH-VALUES
HOLD	I-O	I-O-CONTROL
ID	IDENTIFICATION	IF
IMMATERIAL	IN	INCLUDING
INCREASE	INDEX	INDEXED
INDICATE	INDICATOR	INITIAL
INITIATE	INPUT	INPUT-OUTPUT
INSPECT	INSTALLATION	INTO
INVALID	IS	ITEMS
JUST	JUSTIFIED	KEEP
KEY	KEYS	LABEL
LABELS	LAST	LEADING
LEFT	LENGTH	LESS
LIBRARY	LIMIT	LIMITED
LIMITS	LINAGE	LINE
LINE-COUNTER	LINES	LINK
LINKAGE	LINKED	LIST
LOC	LOCATION	LOCK
LOCKS	LOG	LOW-VALUE
LOW-VALUES	MAND	MANDATORY
MANUAL	MASK	MEMBER

MEMBERS
MEMBERSHIP
MEMORY
MERGE
MESSAGE
MINUS
MIXED
MODE
MODIFY
MODULES
MORE-LABELS
MOVE
MULTIPLE
MULTIPLY
NAME
NAMED
NATIONAL
NATIVE
NEGATIVE
NEXT
NO
NOT
NOTE
NUMBER
NUMERIC
OBJECT-COMPUTER
OCCURRENCE
OCCURS
OF
OFF
OH
OMITTED
ON
ONES
ONLY
OPEN
OPT
OPTIMIZATION
OPTIONAL
OR
ORDER
ORGANIZATION
OTHER
OTHERWISE
OUTPUT
OV
OVERFLOW
OWNER
PA
PAGE
PAGE-COUNTER
PAGES
PERCENT
PERFORM
PERMANENT
PF
PH
PHYSICAL
PHYSICALLY
PIC
PICTURE
PLACEMENT
PLACING
PLUS
POINTER
POINTER-ARRAY
POPULATION
POSITION
POSITIONING
POSITIVE
PRESELECTION
PRINT-SWITCH
PRINTER
PRINTING
PRIOR
PRIVACY
PRIVACY-GROUP
PRIVACY-IDENTIFICATION
PRIVACY-NAME
PRIVACY-RECORD
PROCEDURE
PROCEDURES
PROCEED
PROCESS
PROCESSING
PROGRAM
PROGRAM-ID
PROT
PROTECTED
PROTECTION
PUNCH4
PUNCH6
QUEUE
QUOTE
QUOTES
RANDOM

RD
READ
READER
READY
REAL
REALM
REALM-NAME
REALMS
RECEIVE
RECORD
RECORDING
RECORDS
REDEFINES
REEL
REFERENCE-YEAR
REFERENCES
RELATIVE
RELEASE
REMAINDER
REMARKS
REMOVAL
RENAMES
REORGANIZATION
REPEATED-KEY
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RESTRICTED
RESULT
RETAINING
RETR
RETRIEVAL
RETURN
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROUNDED
RUN
SA
SAME
SCHEMA
SD
SEARCH
SECTION
SECURITY
SEEK
SEGMENT
SEGMENT-LIMIT
SELECT
SELECTION
SELECTIVE
SEND
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SET
SETS
SIEMENS-4004
SIGN
SIGNED
SIZE
SMALLEST
SORT
SORT-MERGE
SORT-TAPE
SORT-TAPES
SORTED
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPANS
SPECIAL-NAMES
STANDARD
STANDARD-1
START
STATUS
STOP
STORAGE
STORE
STRING
STRUCTURE
SUB-QUEUE-1
SUB-QUEUE-2
SUB-QUEUE-3
SUB-SCHEMA
SUBTRACT
SUM

SUPPRESS
SYMBOLIC
SYNC
SYNCHRONIZED
SYSIN
SYSIPT
SYSLST
SYSOPT
SYSOPT-234
SYSOUT
SYSPUNCH
SYSRDR
SYSTEM
TABLE
TALLY
TALLYING
TAPE
TAPES
TEMP
TEMPORARY
TENANT
TERMINAL
TERMINATE
TEXT
THAN
THEN
THROUGH
THRU
TIME
TIMES
TO
TODAYS-DATE
TOP
TRACE
TRACK-AREA
TRACKS
TRAILING
TRANSFORM
TRY
TYPE
UNDER
UNIT
UNITS
UNSTRING
UNTIL
UP
UPDATE
UPON
USAGE
USAGE-MODE
USE
USING
VALUE
VALUES
VARYING
VIA
WHEN
WITH
WITHIN
WITHOUT
WORDS
WORKING-STORAGE
WRITE
WRITE-ONLY
YEAR
ZERO
ZEROES
ZEROS

5 SSL

5.1 Einführung

Mit der Schema-DDL beschreiben Sie die logische Struktur Ihrer Daten; mit der SSL (Storage Structure Language) geben Sie an, welche Speicherstruktur beim physischen Abspeichern der Daten und ihrer logischen Beziehungen entstehen soll. Die Beschreibung der physischen Speicherstruktur ist wahlfrei. Wenn Sie darauf verzichten, benutzt UDS/SQL die Standardwerte. Mit der Beschreibung der Speicherstruktur nehmen Sie Einfluss z.B. auf den Speicherplatzbedarf der Daten und noch mehr auf das Laufzeitverhalten des Systems bei zukünftigen Anwendungen.

Die Gestaltung der Speicherstruktur lässt sich in folgende Arbeitsschritte aufgliedern:

1. Mengengerüst beschreiben
2. Verknüpfung der Sätze bestimmen
3. Lage von Membersätzen, Tabellen und Hashbereichen bestimmen
4. Reorganisationsaufwand für Tabellen festlegen.

Jedem dieser Arbeitsschritte ist im Folgenden ein Abschnitt gewidmet; dort sind auch jeweils die voreingestellten Standardwerte beschrieben. Diesen Abschnitten vorangestellt ist eine Zusammenstellung der Methoden, die UDS/SQL zur physischen Darstellung der logischen Datenstruktur benutzt.

Die verwendete Metasprache ist auf Seite 18 erklärt, die allgemeinen Syntaxregeln sind auf Seite 230 zusammengefasst.

Die SSL übersetzen Sie mit dem SSL-Compiler. Zur Bedienung des SSL-Compilers siehe Handbuch „Aufbauen und Umstrukturieren“, SSL übersetzen).

5.1.1 Methoden zur physischen Darstellung der logischen Datenstruktur

Die physischen Erscheinungsformen für die Gesamtheit

- aller Sätze einer Satzart sind:

DBTT (Database Key Translation Table)	Tabelle, die alle Sätze einer Satzart repräsentiert und die Verbindung von Ownersätzen zu den Tabellen ihrer Set-Occurrences herstellt.
Satz-SEARCH-Key-Tabelle	Tabelle, die alle Sätze einer Satzart repräsentiert.

- aller Sätze einer Set-Occurrence sind:

Adressliste (POINTER-ARRAY) Sort-Key-Tabelle Set-SEARCH-Key-Tabelle	Tabellen, die jeweils alle Membersätze einer Set-Occurrence repräsentieren.
Liste (LIST)	Tabelle, die alle Membersätze einer Set-Occurrence enthält.
Kette (CHAIN)	Enthält Ownersatz und alle Membersätze einer Set-Occurrence.

Sie bestimmen u.a. mit der SSL, ob bzw. wie eine Adressliste, Liste, Kette, Sort-Key-Tabelle oder SEARCH-Key-Tabelle angelegt wird. Eine Beschreibung dieser Elemente finden Sie ab Seite 140.

Dagegen können Sie nicht beeinflussen, dass UDS/SQL eine DBTT anlegt.
Die DBTT ist auf den folgenden Seiten beschrieben.

5.1.2 DBTT (Database Key Translation Table)

Die DBTT stellt die Verbindung her zwischen dem Database-Key-Wert eines Datenbanksatzes und der physischen Seitenadresse dieses Satzes.

Aufbau einer physischen Seitenadresse

Der gesamte Speicherplatz für die Daten der Datenbank ist in Realms aufgeteilt. Jeder einzelne Realm besteht aus Seiten, deren Anzahl beim Datenbankaufbau (siehe Handbuch „Aufbauen und Umstrukturieren“, Datenbank aufbauen) festgelegt wird. In jedem Realm sind die Seiten fortlaufend nummeriert, ebenso sind die Realms nummeriert. UDS/SQL kann also alle Daten in der Datenbank auffinden, wenn es die Nummer der Seite kennt, in der die Daten gespeichert sind und die Nummer des Realm, in dem sich die Seite befindet. Aus diesem Grund sind die Realm-Nummer und die Seitennummer zu einer physischen Seitenadresse zusammengefasst, die UDS/SQL als Zeiger auf die physische Position der Daten benutzt.

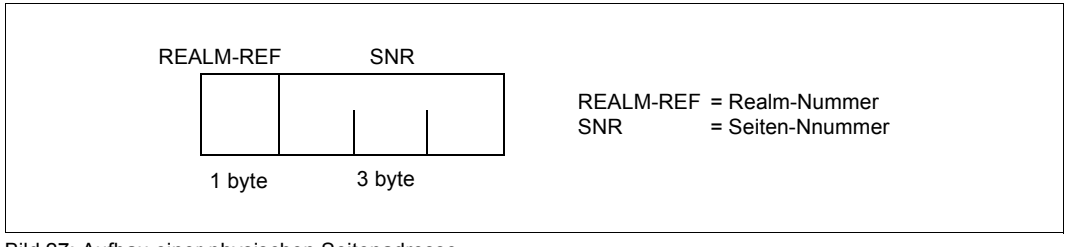


Bild 27: Aufbau einer physischen Seitenadresse

Die Daten in der Datenbank sind insbesondere die von Ihnen definierten Sätze und Tabellen. Die physische Lage von Sätzen und Tabellen kann sich ändern. Bei Lageänderungen aktualisiert UDS/SQL jedoch nicht alle zugehörigen Zeiger.
Zeiger, die immer aktuell sein müssen, heißen Act-Key (Actual-Key), die übrigen heißen Probable Position Pointer (PPP).

Probable Position Pointer (PPP) können mit dem Dienstprogramm BREORG (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BREORG) auf den neuesten Stand gebracht werden.

Aufbau eines Database-Key-Wertes

UDS/SQL kann die physische Adresse von Sätzen und zugehörigen Tabellen immer über ein zusätzliches Kennzeichen finden, das sich während der gesamten Lebensdauer des Satzes in der Datenbank nicht ändert: den Database-Key-Wert.

Eine Satzart besteht aus einer Anzahl von Sätzen, die fortlaufend nummeriert werden. Ebenso trägt jede Satzart eine Nummer. Die Satzfolgenummer (RSQ) und die Nummer der Satzart (REC-REF) bilden zusammen den Database-Key-Wert des Satzes.
Damit ist der Database-Key-Wert ein Kennzeichen, das jeden Satz in der Datenbank eindeutig identifiziert.

Je nachdem, ob die Länge der Datenbankseiten 2048 byte oder 4000 byte bzw. 8096 byte beträgt, sind die Database-Key-Werte unterschiedlich aufgebaut.

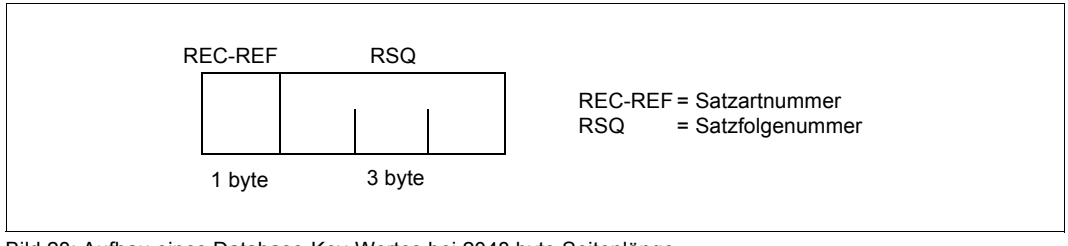


Bild 28: Aufbau eines Database-Key-Wertes bei 2048 byte Seitenlänge

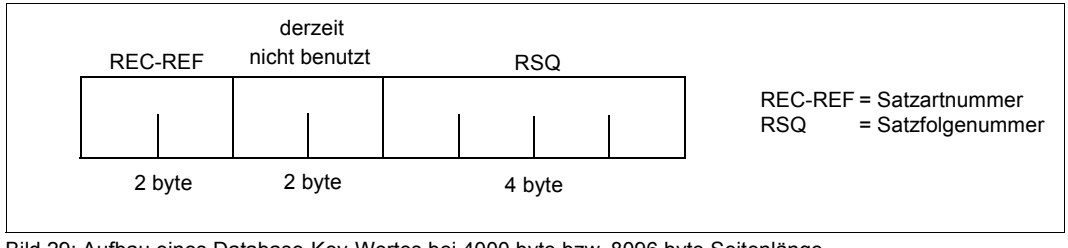


Bild 29: Aufbau eines Database-Key-Wertes bei 4000 byte bzw. 8096 byte Seitenlänge

Für den Wertebereich der Satzartnummer (REC-REF) gilt:

– $1 \leq$ REC-REF $\leq$ 254 bei Datenbanken mit 2048 byte Seitenlänge.
– $1 \leq$ REC-REF $\leq 2^{15}-1$ bei Datenbanken mit 4000/8096 byte Seitenlänge.

Für den Wertebereich der Satzfolgenummer (RSQ) gilt:

– $1 \leq RSQ \leq 2^{24}-1$ bei Datenbanken mit 2048 byte Seitenlänge, falls die Satzart in keinem Set Owner ist.

– $1 \leq RSQ \leq 2^{24}- 2^{16}- 1$ bei Datenbanken mit 2048 byte Seitenlänge, falls die Satzart in irgendeinem Set Owner ist.

– $1 \leq RSQ \leq 2^{31}-1$ bei Datenbanken mit 4000/8096 byte Seitenlänge.

Aufbau der DBTT

Um die physische Lage eines Satzes über den Database-Key-Wert aufzufinden, verwendet UDS/SQL die DBTT.

UDS/SQL richtet für jede Satzart automatisch eine DBTT ein. Zu jedem Satz einer Satzart existiert genau eine Zeile in der zugehörigen DBTT. Eine DBTT-Zeile ist in Spalten unterteilt. In Spalte 0 steht die Seitenadresse des Satzes. Die DBTT einer Satzart, die nicht Ownersatzart ist, besteht insgesamt nur aus Spalte 0.
Der Database-Key-Wert ist nur implizit in der DBTT enthalten. Er wird durch den Ort symbolisiert, an dem die zugehörige physische Seitenadresse gespeichert ist.
Z.B. gilt die dritte Zeile der DBTT für den Satz mit RSQ=3, die neunte Zeile für den Satz mit RSQ=9 usw..

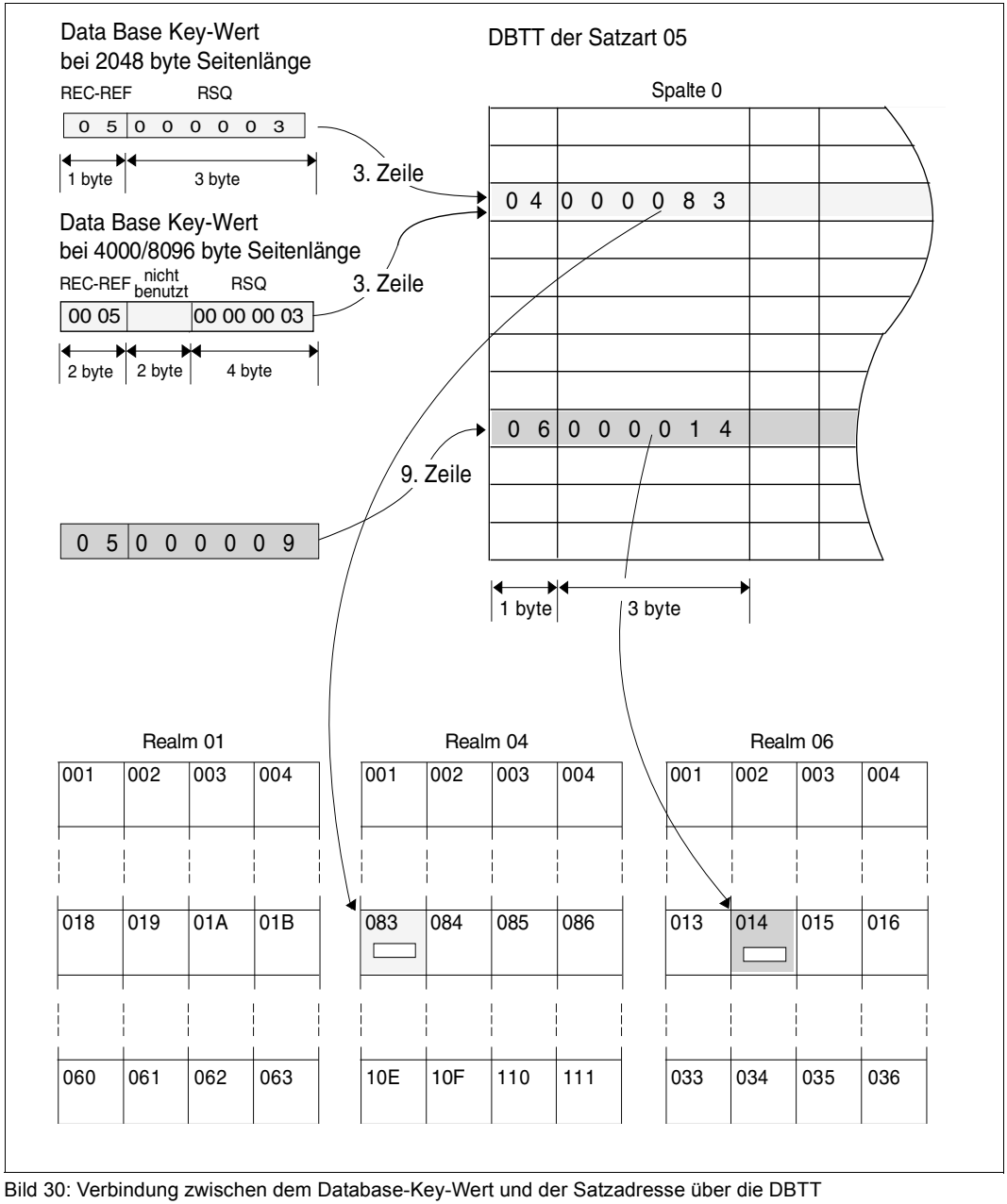


Bild 30: Verbindung zwischen dem Database-Key-Wert und der Satzadresse über die DBTT

Bei der Lageänderung von Sätzen kann nun der Aufwand für die Aktualisierung der Seitenadressen allein auf die DBTT beschränkt bleiben.

Die DBTT einer Ownersatzart enthält außer den Seitenadressen der Ownersätze auch die Seitenadressen aller Tabellen, die zu den Set-Occurrences der Ownersätze gehören. Für diese Tabellen wird der Database-Key-Wert des zugehörigen Ownersatzes zusammen mit der DBTT-Spaltennummer als eindeutiges Kennzeichen verwendet, das sich nicht ändert.

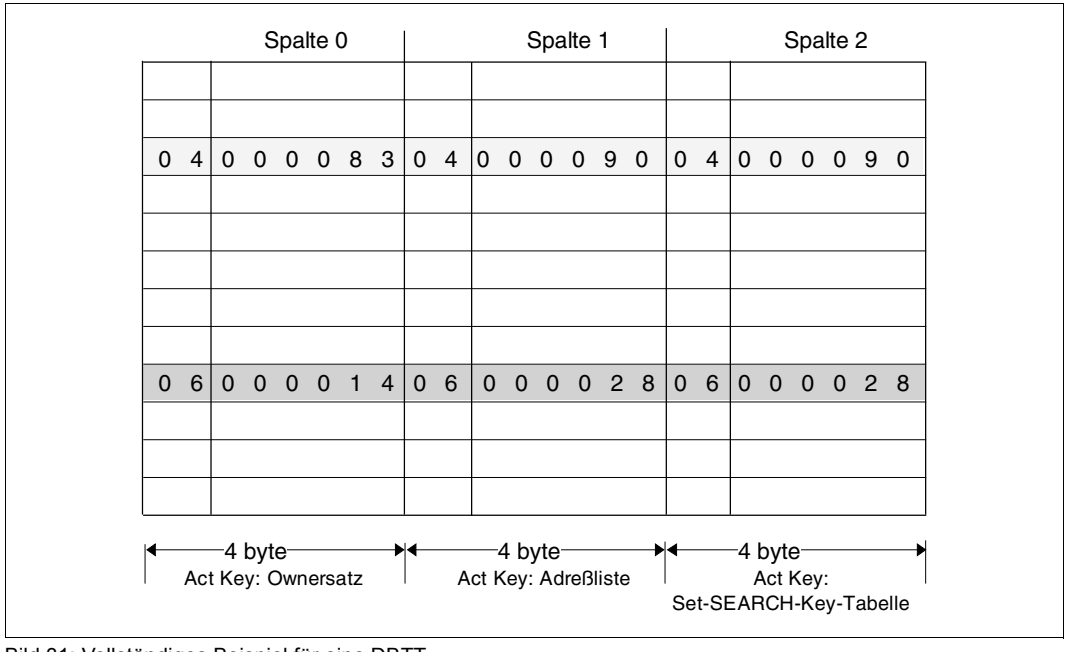


Bild 31: Vollständiges Beispiel für eine DBTT

Es gilt allgemein:

– Die Aktualisierung der Seitenadressen von Sätzen und Tabellen kann auf die DBTT beschränkt bleiben.

– Alle in der DBTT verzeichneten Seitenadressen sind aktuell.

5.2 Mengengerüst beschreiben

5.2.1 Anzahl der Sätze einer Satzart angeben

Den Umfang einer Satzart beschreiben Sie in der DBTT- und der Satz-POPULATION-Klausel. Aus diesen Angaben berechnet UDS/SQL

– den Speicherplatzbedarf für die DBTT,
– die Größe eines Hashbereiches für den Primärschlüssel,
– die Größe von Hashbereichen für Sekundärschlüssel.

Eine Satz-SEARCH-Key-Tabelle wird immer in der Anfangsgröße einer Datenbankseite angelegt und seitenweise erweitert, wenn beim Einspeichern von Sätzen zusätzlicher Bedarf entsteht.

Speicherplatzbedarf für die DBTT

```
DATABASE-KEY-IRANSLATION-IABLE IS ganzzahl [WITHIN realmname]
```

Für jeden Satz muss eine Zeile in der DBTT der zugehörigen Satzart zur Verfügung stehen. Mit *ganzzahl* geben Sie die Anzahl der Sätze, die Sie für die Satzart erwarten, und somit auch die Anzahl der DBTT-Zeilen an:

– Bei einer Datenbank mit 2048 byte Seitenlänge können Sie für *ganzzahl* maximal den Wert 16 777 215 (=2^{24}-1) angeben. Wenn die betreffende Satzart Ownersatzart in einem Set ist, dürfen Sie für *ganzzahl* maximal den Wert 16 711 679 angeben.

– Bei einer Datenbank mit 4000 byte oder 8096 byte Seitenlänge können Sie für *ganzzahl* maximal den Wert 2^{31}-1 angeben.

Die DBTT besteht aus genau einer Spalte, wenn die Satzart keine Ownersatzart ist. Im anderen Fall kommt eine Spalte hinzu für jeden Verweis auf

– eine Adressliste (MODE IS POINTER-ARRAY),

– eine Liste (MODE IS LIST),

– eine Sort-Key-Tabelle (MODE IS CHAIN, ORDER IS SORTED INDEXED),

– eine Set-SEARCH-Key-Tabelle.

Aus der Zeilen- und Spaltenanzahl berechnet UDS/SQL den Speicherplatzbedarf der DBTT und reserviert die benötigte Anzahl von Seiten. Da für eine DBTT Seiten oder DBTT-Extents immer nur komplett reserviert werden, kann sie mehr Zeilen aufnehmen, als Sie mit *ganzzahl* angeben.

Andererseits legt UDS/SQL pro DBTT immer nur so viele DBTT-Seiten an, dass je nach Seitengröße der Datenbank maximal 2^{24}-1 bzw. 2^{24}- 2^{16}- 1 (=16711679) bzw. 2^{31}-1 Einträge pro DBTT verwaltet werden können. Da eine DBTT-Basis immer nur in ganzen Seiten und ein DBTT-Extent immer nur in DBTT-Extentgröße (128 PAM-Seiten) angelegt werden, wird bei den Maximalangaben in der Regel ein Verschnitt entstehen. Dieser Verschnitt wird nicht genutzt.

Darüber hinaus können Sie die DBTT mit dem Dienstprogramm BREORG vergrößern (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BREORG).

Im Standardfall reserviert UDS/SQL eine Seite für eine DBTT.

Wenn Sie für die Satzart einen Sekundärschlüssel zur Umrechnung durch ein Hashverfahren definiert haben, wirkt sich die Größenangabe für die DBTT auch auf die Anfangsgröße des Hashbereichs aus (siehe Abschnitt „Größe eines Hashbereiches für den Primärschlüssel“ auf Seite 134).

Größe eines Hashbereiches für den Primärschlüssel

```
POPULATION IS {ganzzahl WITHIN realmname},...
```

Der Hashbereich für den Primärschlüssel einer Satzart ist auf mehrere Realms verteilt, wenn Sie mit der Schema-DDL die Sätze der Satzart auf mehrere Realms verteilen (siehe Abschnitt „Verteilung von Sätzen auf Realms bestimmen“ auf Seite 105) und wenn die Satzart nicht Membersatzart einer verteilbaren Liste ist. Der Hashbereich für einen indirekten CALC über einer verteilbaren Liste liegt für alle Sätze im Tabellenrealm (Informationen zu verteilbaren Listen finden Sie auf Seite 145).

Mit *ganzzahl* geben Sie die Anzahl der Sätze an, die in dem Realm *realmname* gespeichert werden sollen. UDS/SQL berechnet daraus die Mindestanzahl von Seiten, die zur Speicherung der Sätze (siehe Abschnitt „Direkte CALC-Seite“ auf Seite 208) bzw. der Satzadressen (siehe Abschnitt „Indirekte CALC-Seite“ auf Seite 211) im jeweiligen Realm nötig sind. UDS/SQL reserviert daraufhin in jedem Realm die Anzahl von Seiten, die der jeweils nächsthöheren Primzahl entspricht.

Die Größe eines Hashbereichs über einer verteilbaren Liste wird aus der Summe der POPULATION-Werte der beteiligten Realms bestimmt. Für die tatsächliche Verteilung der Sätze auf die an der verteilbaren Liste beteiligten Realms hat die POPULATION-Angabe keine Bedeutung.

Ein Hashbereich wird durch Überlaufseiten erweitert, wenn das Hashverfahren einer Seite mehr Daten zuweist, als dieser aufnehmen kann. Die überlaufende Seite ist mit der Überlaufseite über Act-Keys verkettet.

Die Qualität eines Hashverfahrens ist daran zu messen, wie gleichmäßig es die Daten über den Hashbereich verteilt, d.h. wie viele Überlaufseiten es erzeugt.

Ein Hashbereich ist nachträglich mit dem Dienstprogramm BREORG (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BREORG) erweiterbar.

Standardmäßig richtet UDS/SQL einen Hashbereich in der Größe einer Seite ein.

Beispiel

DDL:

```
RECORD NAME IS ARTIKEL
    LOCATION MODE IS CALC
    WITHIN KLEIDUNG, HAUSHALT, ..... AREA-ID .....
    .
    .
    .
```

SSL:

```
RECORD NAME IS ARTIKEL
    POPULATION IS 400 WITHIN KLEIDUNG, 100 WITHIN HAUSHALT, .....
```

Größe des Hashbereichs für einen Sekundärschlüssel

DATABASE-KEY-TRANSLATION-TABLE IS *ganzzahl*

Mit *ganzzahl* geben Sie die Anzahl von Sätzen der Satzart an:

– Bei einer Datenbank mit 2048 byte Seitenlänge können Sie für *ganzzahl* maximal den Wert 2^{24}-1 angeben. Wenn die betreffende Satzart Ownersatzart in einem Set ist, dürfen Sie für *ganzzahl* maximal den Wert 16 711 679 angeben.

– Bei einer Datenbank mit 4000 byte oder 8096 byte Seitenlänge können Sie für *ganzzahl* maximal den Wert 2^{31}-1 angeben.

UDS/SQL berechnet daraus nicht nur die Größe der DBTT, sondern auch die Mindestzahl von Seiten, die zur gestreuten Speicherung der Satzadressen nötig sind (siehe Abschnitt „Indirekte CALC-Seite“ auf Seite 211).
Daraufhin reserviert UDS/SQL die Anzahl von Seiten, die der nächsthöheren Primzahl entspricht.

Ein Hashbereich wird durch Überlaufseiten erweitert, wenn das Hashverfahren einer Seite mehr Satzadressen zuweist, als dieser aufnehmen kann. Die überlaufende Seite ist mit der Überlaufseite über Act-Keys verkettet.

Ein Hashbereich ist nachträglich mit dem Dienstprogramm BREORG erweiterbar (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BREORG).

Im Standardfall richtet UDS/SQL jeden Hashbereich für einen Sekundärschlüssel in der Größe einer Seite ein.

5.2.2 Größe der Set-Occurrences eines Set angeben

```
POPULATION IS ganzzahl
```

Die Größe der Set-Occurrences müssen Sie in folgenden Fällen angeben:

– Sie wollen Tabellen und Membersätze in die Nähe der zugehörigen Ownersätze speichern.

– Sie wollen den Aufwand für nachträgliche Tabellenerweiterungen minimieren.

– Sie wollen im SYSTEM-Set die Größe eines Hashbereiches für Sekundärschlüssel angeben, um Überlaufseiten zu vermeiden.

Mit *ganzzahl* geben Sie eine durchschnittliche Set-Occurrence-Population an: Sie sollten den Wert wählen, der auf die Mehrheit der Set-Occurrences zutrifft.

Aus dieser Angabe berechnet UDS/SQL

– den Speicherplatzbedarf für Adresslisten, Listen, Sort-Key-Tabellen und Set-SEARCH-Key-Tabellen, falls der Set kein SYSTEM-Set ist,

– den Speicherplatzbedarf für Membersätze der Set-Ocurrences, falls der Set kein SYSTEM-Set ist,

– die Größe von Hashbereichen für Sekundärschlüssel in einem SYSTEM-Set.

Den für eine Set-Occurrence berechneten Speicherplatz reserviert UDS/SQL beim Abspeichern des Ownersatzes im nächsten freien Platz, der auf den Ownersatz folgt.

Der Standardwert für *ganzzahl* ist 0. In diesem Fall wird für die Tabellen und Membersätze kein Platz und für Hashbereiche jeweils eine Seite reserviert.

Speicherplatzbedarf für Adressliste, Liste, Sort-Key-Tabelle und Set-SEARCH-Key-Tabelle

Jede der genannten Tabellen repräsentiert bzw. enthält alle Membersätze einer Set-Occurrence. Jedem Membersatz entspricht eine Zeile in den zugehörigen Tabellen. D.h. mit *ganzzahl* bestimmen Sie die Zeilenanzahl je Tabelle der Set-Occurrence.

Aus der Zeilenanzahl und Zeilenlänge berechnet UDS/SQL den Speicherplatzbedarf der Tabellen.

Beim Abspeichern eines Ownersatzes reserviert UDS/SQL für jede Tabelle, die zu seiner Set-Occurrence gehört, Speicherplatz in folgender Größe:

– den berechneten Speicherplatz, wenn dieser kleiner als eine Seite ist
– eine Seite, wenn der errechnete Speicherplatz größer als eine Seite ist

Bei SYSTEM-Sets reserviert UDS/SQL für jede Tabelle genau eine Datenbankseite, unabhängig von der POPULATION-Klausel.

Tabellenerweiterungen

UDS/SQL erweitert Adresslisten, Listen, Sort-Key-Tabellen und Set-SEARCH-Key-Tabellen automatisch, wenn beim Einspeichern von Membersätzen zusätzlich Bedarf entsteht. Um unnötig häufige Tabellenerweiterungen zu vermeiden, können Sie deren Umfang durch eine Zusatzangabe in der Set-POPULATION-Klausel steuern.

`POPULATION IS` *`ganzzahl-1`* `INCREASE IS` *`ganzzahl-2`*

Mit *ganzzahl-2* geben Sie die Anzahl der Tabellenzeilen an, die bei den ersten beiden Tabellenerweiterungen angefügt werden soll. Ab der zweiten Erweiterung werden *ganzzahl-2* * 2^n (n⌉ Anzahl der Erweiterungen) Tabellenzeilen angefügt, bis kein Platz mehr für eine zusätzliche Tabellenzeile vorhanden ist. Falls die Seite noch andere Daten enthält, die die Tabelle an einer Ausdehnung hindern, geschieht eine nachfolgende Tabellenerweiterung durch Umspeichern der Tabelle. In diesem Fall sucht UDS/SQL eine Seite, die die Tabelle mit der gewünschten Erweiterung aufnehmen kann.

Für Tabellenerweiterungen über die erste Tabellenseite hinaus ist *ganzzahl-2* nicht mehr wirksam (siehe Abschnitt „Reorganisationsaufwand für Tabellen festlegen“ auf Seite 170).

Standardwert für *ganzzahl-2* ist 1.

Speicherplatzbedarf für Membersätze

Hier sind nur Membersätze gemeint, die nicht in eine Liste gespeichert werden. Beim Abspeichern des Ownersatzes reserviert UDS/SQL Platz für dessen Membersätze, wenn Sie die Membersatzart mit PLACEMENT OPTIMIZATION definiert haben (siehe Abschnitt „PLACEMENT OPTIMIZATION“ auf Seite 162). In diesem Fall zieht UDS/SQL die Angabe der Set-Occurrence-Population heran, um den Speicherplatzbedarf für die Membersätze zu berechnen.

Größe des Hashbereiches für einen Sekundärschlüssel

Nur bei SYSTEM-Sets können Sie den Sekundärschlüssel zur Umrechnung durch ein Hashverfahren heranziehen. In diesem Fall berechnet UDS/SQL aus der Angabe der Set-Occurrence-Population die Mindestanzahl von Seiten, die zum gestreuten Speichern der Satzadressen nötig sind (siehe Abschnitt „Indirekte CALC-Seite“ auf Seite 211).
UDS/SQL reserviert daraufhin die Anzahl von Seiten, die der nächsthöheren Primzahl entspricht.

Sollten einige Seiten infolge einer ungleichmäßigen Datenverteilung durch das Hashverfahren überlaufen, so legt UDS/SQL automatisch Überlaufseiten an.

Ein Hashbereich ist nachträglich mit dem Dienstprogramm BREORG erweiterbar (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BREORG).

Im Standardfall richtet UDS/SQL einen Hashbereich in der Größe einer Seite ein.

5.2.3 Übersicht über die Anfangsgrößen von Speicherplatzreservierungen

Art der Daten	**Settyp**	**reservierter Speicherplatz**
Sätze mit PLACEMENT OPTIMIZATION	nicht	entsprechend *ganzzahl* in Set-POPULATION-Klausel
Liste Adressliste Sort-Key-Tabelle	SYSTEM	entsprechend *ganzzahl* in Set-POPULATION-Klausel; höchstens eine Seite
Set-SEARCH-Key-Tabelle	SYSTEM	eine Seite
Satz-SEARCH-KEY-Tabelle	-	
Hashbereich für Set-SEARCH-Key	SYSTEM	entsprechend *ganzzahl* in Set-POPULATION-Klausel; mindestens eine Seite
Hashbereich für Satz-SEARCH-Key	-	entsprechend *ganzzahl* in DBTT-Klausel; mindestens eine Seite
Hashbereich für Primärschlüssel		entsprechend *ganzzahl* in Satz-POPULATION-Klausel; mindestens eine Seite
DBTT		entsprechend *ganzzahl* in DBTT-Klausel; mindestens eine Seite

Tabelle 7: Anfangsgrößen für Speicherplatzreservierungen

5.3 Verknüpfung der Sätze bestimmen

5.3.1 Speicherungsart der Set-Occurrences bestimmen

Die Verknüpfung von Membersätzen zu einer Set-Occurrence kann UDS/SQL durch drei verschiedene Speicherungsarten für Set-Occurrences realisieren. Diese sind:

- Adressliste (POINTER-ARRAY)
- Liste (LIST)
- Kette (CHAIN)

Zur näheren Erläuterung der drei Speicherungsarten dient in diesem Abschnitt folgendes Beispiel:

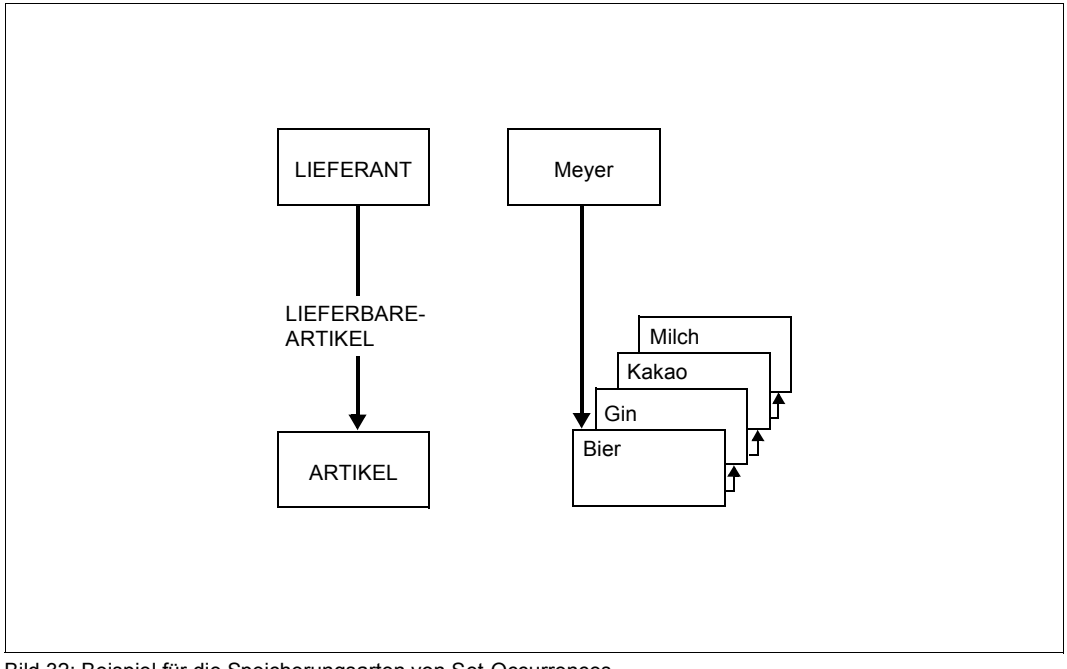


Bild 32: Beispiel für die Speicherungsarten von Set-Occurrences

Speicherung einer Set-Occurrence als Adressliste

MODE IS POINTER-ARRAY

Wenn Sie für einen Set MODE IS POINTER-ARRAY definieren, so verknüpft UDS/SQL die Membersätze einer Set-Occurrence über eine Tabelle, die als Adressliste bezeichnet wird.

Die Tabelle enthält die Zeiger zu jedem Membersatz der Set-Occurrence. Ein Zeiger setzt sich aus der RSQ des Membersatzes und dem Probable Position Pointer (PPP) der Seite zusammen, in der der Membersatz gespeichert ist (siehe Seite 221, Tabellenzeile).

Das weitere Aussehen der Adressliste hängt davon ab, welche ORDER-Klausel Sie mit der Schema-DDL für den Set definiert haben:

– ORDER IS LAST/FIRST/NEXT/PRIOR/IMMATERIAL oder
SORTED INDEXED BY DATABASE-KEY

  Die Adressliste enthält nur die Zeiger zu den Membersätzen der Set-Occurrence.

– ORDER IS SORTED INDEXED BY DEFINED KEYS

  Die Adressliste enthält außer den Zeigern zu den Membersätzen der Set-Occurrence die zugehörigen Schlüsselwerte des Primärschlüssels.

– ORDER IS SORTED

  Diese Angabe ist bei einer Adressliste nicht erlaubt.

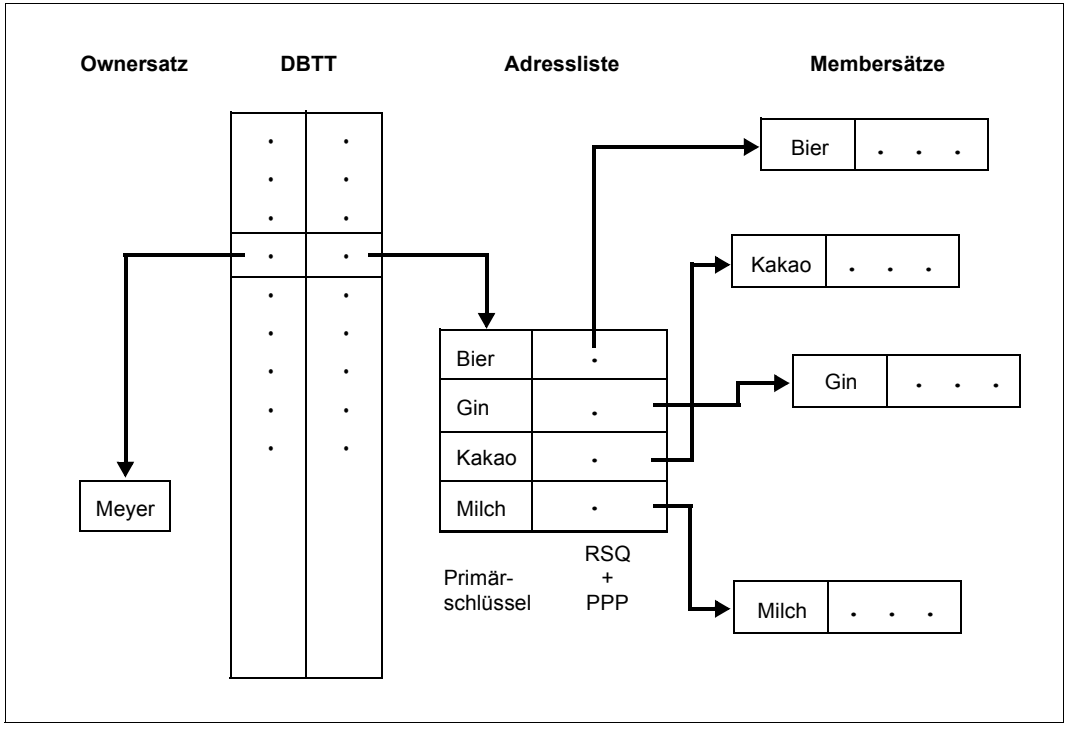


Bild 33: Set-Occurrence als Adressliste gespeichert

Nimmt eine Adressliste mehrere Seiten in Anspruch so sind alle Seiten zweifach über Act-Keys miteinander verknüpft.
Im Fall von ORDER IS SORTED INDEXED erhält eine Adressliste dann zusätzlich übergeordnete Tabellenstufen. Jede übergeordnete Stufe ist genauso aufgebaut wie die unterste Stufe, enthält jedoch nur den jeweils letzten Tabelleneintrag der Seiten, die die nächstniedrigere Stufe enthalten. An die Stelle des Probable Position Pointer (PPP) auf Membersätze tritt der Act-Key der Seite, aus der der Tabelleneintrag stammt (siehe Seite 225, Bild 59).

Zusätzlicher Zeiger vom Owner auf seine Adressliste

```
MODE IS POINTER-ARRAY.....WITH PHYSICAL LINK
```

Die Verbindung eines Ownersatzes mit der Adressliste seiner Membersätze ist standardmäßig nur über die DBTT hergestellt. Sie können den Umweg über die DBTT und damit einen Seitenzugriff sparen, wenn Sie durch die Angabe WITH PHYSICAL LINK im Ownersatz einen Zeiger auf die Adressliste anlegen.

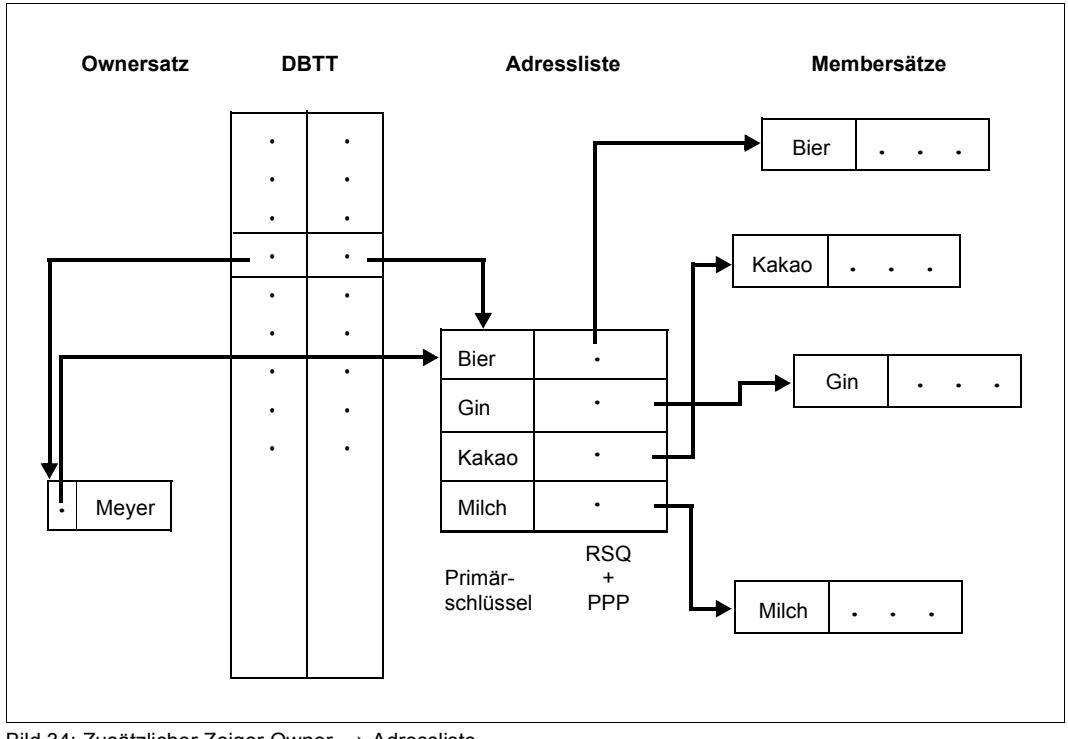


Bild 34: Zusätzlicher Zeiger Owner ⟶ Adressliste

In SYSTEM-Sets übernimmt ein Ankersatz die Funktion eines Ownersatzes und einer DBTT gleichzeitig. In diesem Fall ist die Angabe verboten.

Speicherung einer Set-Occurrence als Liste

```
MODE IS LIST
```

Wenn Sie für einen Set MODE IS LIST definieren, so speichert UDS/SQL die Membersätze einer Set-Occurrence in eine Tabelle, die als Liste bezeichnet wird. Die physische Reihenfolge der Sätze entspricht der logischen Reihenfolge, die Sie mit der ORDER-Klausel definiert haben.

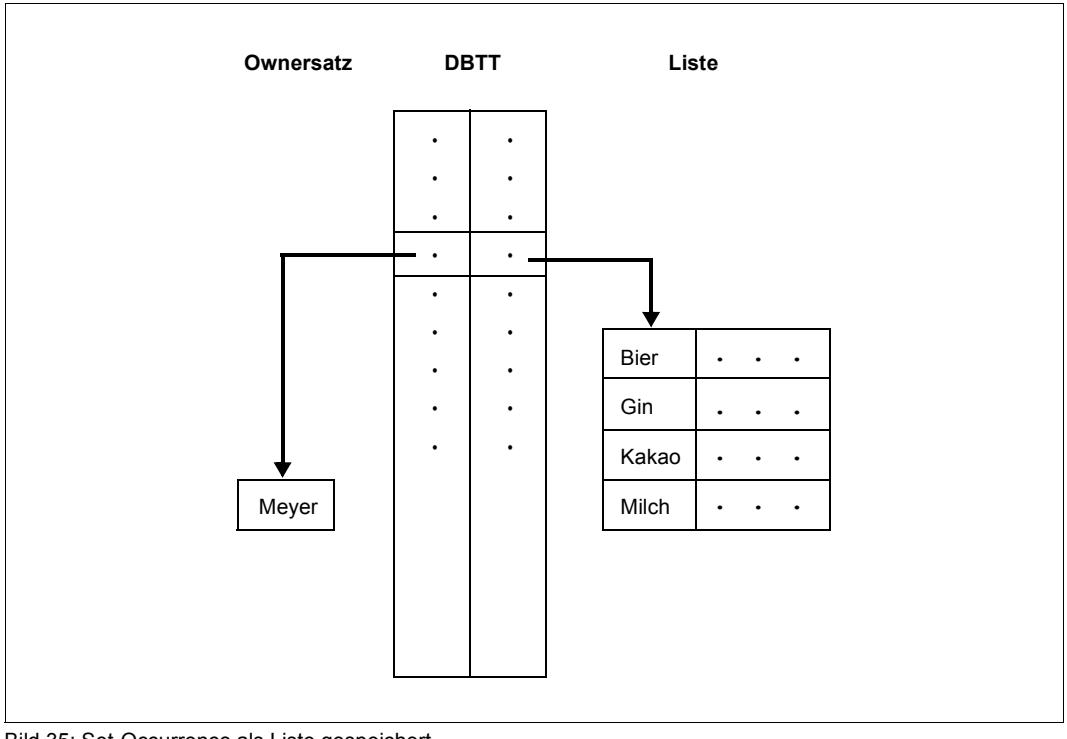


Bild 35: Set-Occurrence als Liste gespeichert

Nimmt eine Liste mehrere Seiten in Anspruch, so sind alle Seiten zweifach über Act-Keys miteinander verknüpft.

Im Fall ORDER IS SORTED INDEXED erhält eine Liste dann zusätzlich übergeordnete Tabellenstufen. Die übergeordneten Stufen haben dieselbe Form wie die einer Adressliste.

Die physische Lage von Sätzen kann nur einmal bestimmt werden:

– Ist eine Satzart Member in mehreren Sets, so kann nur einer der Sets mit MODE IS LIST erklärt werden.

– Sätze können nicht gleichzeitig mit MODE IS LIST und PLACEMENT OPTIMIZATION platziert werden (siehe Abschnitt „PLACEMENT OPTIMIZATION“ auf Seite 162).

– Platzieren Sie die Sätze einer Satzart sowohl über LOCATION MODE IS CALC als auch über MODE IS LIST, so werden die Sätze in Listen gespeichert und der Hashbereich besteht aus indirekten CALC-Seiten

Verteilbare Liste:

Eine Liste kann über mehrere Realms verteilt werden ("verteilbare Liste"). Hierzu sind folgende Voraussetzungen notwendig:

– In der SSL-Set-Deklaration müssen Sie MODE IS LIST angeben (siehe Seite 250).

– In der DDL-Set-Deklaration müssen Sie OWNER IS SYSTEM und ORDER IS SORTED INDEXED angeben (siehe Seite 239).

– In der DDL-Satz-Deklaration müssen Sie in der WITHIN-Klausel der Deklaration der Membersatzart mehrere Realms angeben (siehe Seite 105 bzw. Seite 236).

Die Verteilung auf mehrere Realms betrifft nur die Stufe-0-Seiten, in denen sich die Sätze selbst befinden.

Der Index der Liste (Seiten mit Stufen > 0) liegt in einem Realm („Tabellenrealm“).

Beim Abspeichern von Sätzen im DBH werden freie Seiten in dem bevorzugten Realm („Preferred-Realm“) gesucht. Der Preferred-Ralm kann mit der UDS-Online-Utility neu gesetzt bzw. verändert werden (siehe Handbuch „Sichern, Informieren und Reorganisieren“).

Außerdem gelten bei Listen die folgenden Einschränkungen:

MODE IS LIST dürfen Sie nur angeben, wenn Sie die Set-Mitgliedschaft mit der Schema-DDL als MANDATORY AUTOMATIC definiert haben (siehe Abschnitt „Art der Set-Mitgliedschaft von Membersätzen definieren“ auf Seite 75).

MODE IS LIST dürfen Sie in folgenden Fällen nicht angeben:

- Die Satzlängen (einschl. Zeigern, siehe Seite 217, SCD) betragen mehr als
  - 993 byte in einer Datenbank mit 2048 byte Seitenlänge (2-Kbyte-Seitenformat)
  - 1963 byte in einer Datenbank mit 4000 byte Seitenlänge (4-Kbyte-Seitenformat)
  - 4011 byte in einer Datenbank mit 8096 byte Seitenlänge (8-Kbyte-Seitenformat)

  (Mindestens zwei Sätze müssen in eine Liste passen, was in den genannten Fällen nicht mehr gegeben ist.)

- Die Membersatzart enthält ein Feld variabler Länge.
- Für die Membersatzart des Set wurden in der Schema-DDL in der WITHIN-Klausel der RECORD-Deklaration mehrere Realms angegeben, die Ownersatzart des Set ist Member einer verteilbaren Liste und in der MODE IS LIST-Anweisung ist DETACHED ohne WITHIN-Klausel angegeben.
- Die Ownersatzart des Set ist Member einer verteilbaren Liste und in der MODE IS LIST-Anweisung ist ATTACHED angegeben.
- Die Reihenfolge der Sätze innerhalb der Set-Occurrence ist mit ORDER IS SORTED (ohne INDEXED) festgelegt.
- Sie wollen die Membersätze komprimiert abspeichern (siehe Abschnitt „Die Sätze einer Satzart komprimiert speichern“ auf Seite 174).

Zusätzlicher Zeiger vom Owner auf seine Liste

MODE IS LIST.....WITH PHYSICAL LINK

Die Verbindung eines Ownersatzes mit der Liste seiner Membersätze ist standardmäßig nur über die DBTT hergestellt. Sie können den Umweg über die DBTT und damit einen Seitenzugriff sparen, wenn Sie durch die Angabe WITH PHYSICAL LINK im Ownersatz einen Zeiger auf die Liste anlegen.

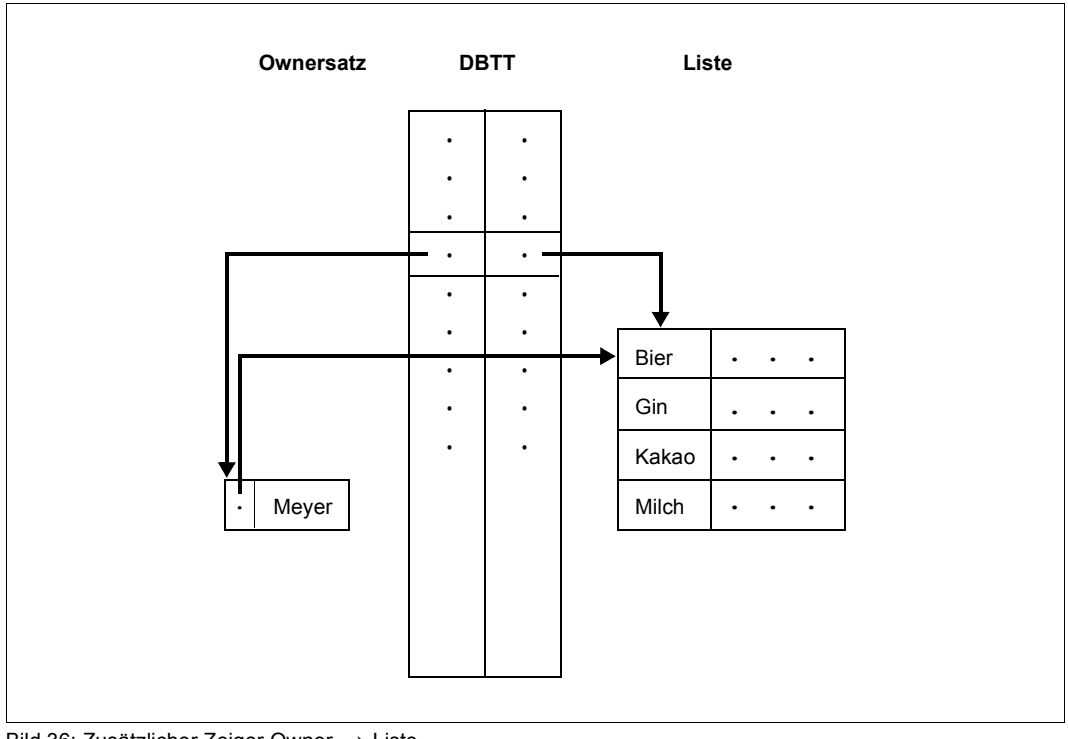


Bild 36: Zusätzlicher Zeiger Owner ⟶ Liste

In SYSTEM-Sets übernimmt ein Ankersatz die Funktion eines Ownersatzes und einer DBTT gleichzeitig. In diesem Fall ignoriert der Compiler die Angabe und informiert den Benutzer durch eine Warnung.

Speicherung einer Set-Occurrence als Kette

MODE IS CHAIN

Wenn Sie für einen Set MODE IS CHAIN definieren, so wird jede Occurrence des Set als eine in sich geschlossene Kette von Sätzen gespeichert. Die Kette besteht aus dem Ownersatz und allen zugehörigen Membersätzen der Set-Occurrence. Die Verkettung der Sätze wird über Zeiger so hergestellt, dass die Membersätze in der Kette die Reihenfolge einnehmen, die Sie in der ORDER-Klausel der Schema-DDL für den Set definiert haben. Der Ownersatz nimmt den Platz zwischen dem letzten und dem ersten Membersatz ein.

Die Zeiger bestehen aus dem Database-Key-Wert des in der Kette nachfolgenden Satzes und dem Probable Position Pointer (PPP) der Seite, in der dieser Satz gespeichert ist.

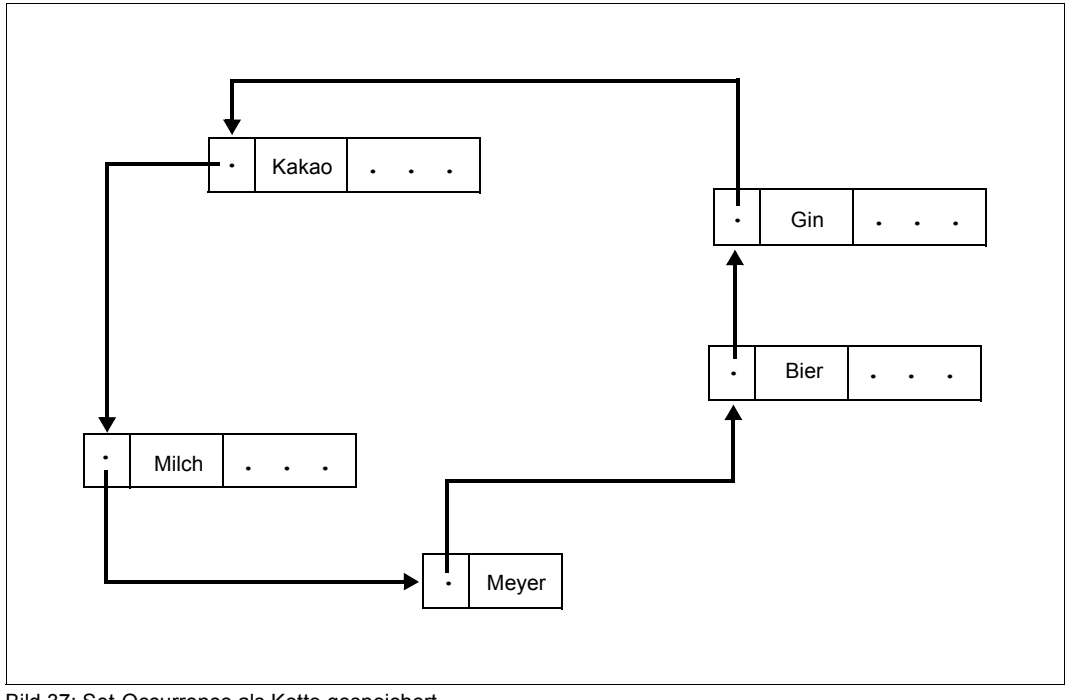


Bild 37: Set-Occurrence als Kette gespeichert

Die Kette ist die einzige Speicherungsart, die Sie anwenden dürfen, wenn der Set im Schema mit ORDER IS SORTED (ohne INDEXED) definiert ist.
Im Falle ORDER IS SORTED INDEXED entsteht zusätzlich eine Tabelle als Zugriffspfad

für Direktzugriff.
Diese Tabelle heißt Sort-Key-Tabelle. Sie hat denselben Aufbau wie eine mehrstufige Adressliste.

Zusätzliche Rückwärtsverkettung für Kette

```
MODE IS CHAIN LINKED TO PRIOR
```

Zusätzlich zur standardmäßigen Vorwärtsverkettung können Sie die Sätze einer Kette in umgekehrter Reihenfolge verketten. Wenn Sie für eine Kette LINKED TO PRIOR angeben, wird in jedem Satz der Kette zusätzlich ein Zeiger zum logisch vorangehenden Satz aufgenommen. Der Zeiger besteht analog zur Vorwärtsverkettung aus dem Database-Key-Wert des vorangehenden Satzes und dem Probable Position Pointer (PPP) der Seite, in der dieser Satz gespeichert ist.

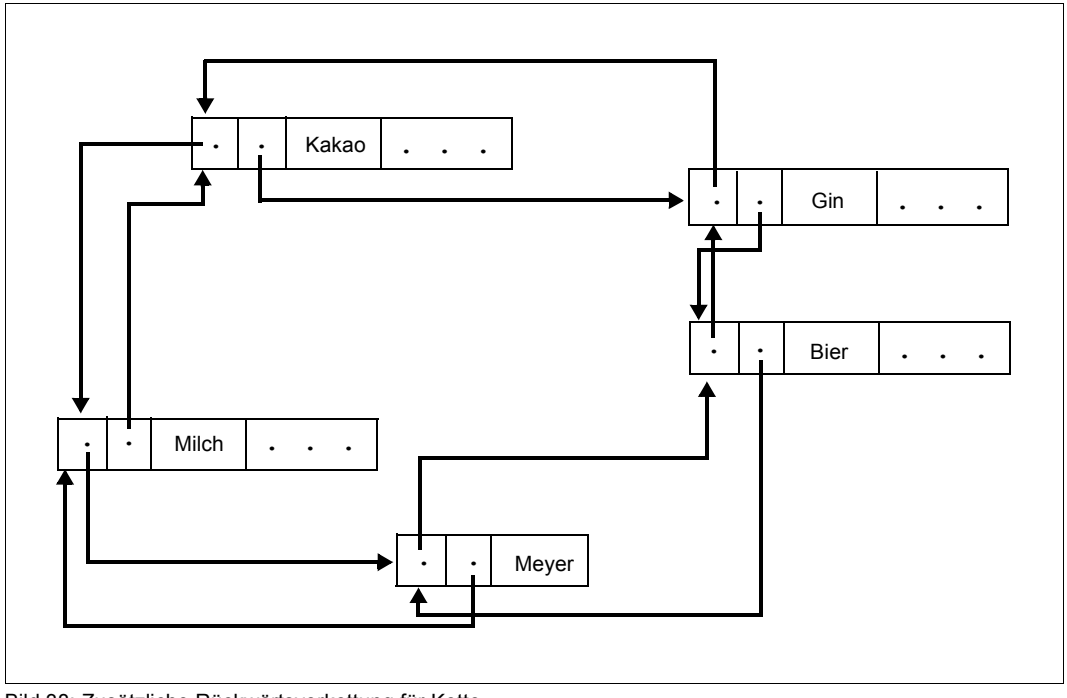


Bild 38: Zusätzliche Rückwärtsverkettung für Kette

Die Rückwärtsverkettung ist wichtig bei großen Set-Occurrences, wenn Sie häufig den Vorgänger eines Satzes suchen müssen.

5.3.2 Bewertung von Adressliste, Liste und Kette

Die Zeit für die Ausführung eines Programms hängt von der Speicherungsart der Set-Occurrences ab.

In diesem Abschnitt werden Wiedergewinnungs- und Änderungsoperationen bei Adressliste, Liste und Kette verglichen. Die Wiedergewinnungsoperationen sind unterteilt in sequenzielle und direkte Zugriffe. Von den Änderungsoperationen werden Einfügen und Löschen behandelt.

Adressliste

- Sequenzieller Zugriff

  Die Adressliste muss im Hauptspeicher vorhanden sein. Das ist meist der Fall, wenn bereits über den Set auf einen Membersatz der Set-Occurrence zugegriffen wurde. Anschließend muss jeweils ein Eintrag in der Adressliste gelesen werden, bevor der zugehörige Satz gelesen werden kann. Dabei hängt die Anzahl von Plattenzugriffen davon ab, wie die Sätze über die Seiten verteilt sind. Die Anzahl der Plattenzugriffe lässt sich durch eine optimierte Platzierung der Sätze reduzieren (siehe Abschnitte „Natürliche Optimierung“ auf Seite 161 und „PLACEMENT OPTIMIZATION“ auf Seite 162).

- Direktzugriff

  – ORDER IS SORTED INDEXED: Alle Stufen der Adressliste müssen im Hauptspeicher sein. Die höheren Stufen werden dazu verwendet, die Nummer der Seite auszuwählen, in der der Eintrag des gesuchten Satzes liegt. Für jede Tabellenstufe ist ein Plattenzugriff notwendig.

  – ORDER IS LAST/FIRST/NEXT/PRIOR: UDS/SQL muss die Set-Occurrence sequenziell absuchen, bis der gewünschte Satz gefunden ist. Dies kann relativ viele Plattenzugriffe bedeuten.

- Einfügen

  In der Adressliste muss der Einfügungsort für einen Tabelleneintrag bestimmt werden. Dies geschieht im Falle ORDER IS SORTED INDEXED mit einem Direktzugriff.
  Bei ORDER IS LAST/FIRST kann UDS/SQL direkt auf die letzte bzw. erste Tabellenseite zugreifen. Bei ORDER IS NEXT/PRIOR wird der Einfügungsort entweder durch die Currency-Information oder durch sequenzielles Lesen der Adressliste gefunden. Zum Schreiben des Satzes ist im Allgemeinen ein weiterer Plattenzugriff nötig.

- Löschen

  Außer dem Satz muss der zugehörige Eintrag aus der Adressliste gelöscht werden.

- Ergebnis

  MODE IS POINTER-ARRAY führt zu schnellen sequenziellen und direkten Zugriffen und Änderungsoperationen. Das Zeitverhalten ist weitgehend unabhängig von der Setordnung und Größe der Set-Occurrences.

Liste

- Sequenzieller Zugriff

  MODE IS LIST bündelt die Sätze auf einen zusammenhängenden Speicherbereich. Es ist deshalb die Speicherungsart, die die schnellste sequenzielle Verarbeitung bietet. Die Anzahl der Plattenzugriffe bei der Verarbeitung größerer Satzmengen hängt von der Satzlänge ab.

- Direktzugriff

  Wenn Sie ORDER IS SORTED INDEXED definiert haben, müssen alle Stufen der Liste im Hauptspeicher sein. Die höheren Stufen werden dazu verwendet, die Nummer der Seite auszuwählen, in der sich der Satz befindet. Für jede unterste Tabellenstufe ist meist ein Plattenzugriff notwendig. Die Anzahl der Tabellenstufen hängt von der Satzlänge ab.
  Wenn Sie nicht ORDER IS SORTED INDEXED definiert haben, muss UDS/SQL die Set-Occurrence vom Beginn an durchsuchen. Die Anzahl der dafür notwendigen Plattenzugriffe hängt stark von der Satzlänge und der Größe der Set-Occurrence ab.

- Einfügen

  Die Stelle, an die der Satz eingesetzt werden soll, findet UDS/SQL bei ORDER IS SORTED INDEXED durch einen Direktzugriff. Ansonsten kann UDS/SQL die Stelle durch maximal zwei Plattenzugriffe finden.
  Das Einfügen eines Satzes kann bewirken, dass UDS/SQL eine Anzahl von Sätzen umspeichern muss.

- Löschen

  UDS/SQL muss im Allgemeinen nur die Seite ändern, aus der der Satz zu löschen ist.

- Ergebnis

  MODE IS LIST bietet die schnellste sequenzielle Zugriffsmethode. Bewirkt eine Änderungsoperation eine Änderung der Satzreihenfolge, so kann das zu physischen Verschiebungen der Sätze führen. In SEARCH-Key-Tabellen, die auf diese Sätze verweisen, stimmen dann die Probable Position Pointer (PPP) nicht mehr.
  Das Zugriffsverhalten hängt stark von der Ordnung und der Größe der Set-Occurrences ab.

Kette

- Sequenzieller Zugriff

  Wenn Sie die Sätze gemäß ihrer logischen Reihenfolge verarbeiten, ist maximal für jeden Membersatz ein Plattenzugriff erforderlich. Diese Anzahl von Plattenzugriffen lässt sich durch eine optimierte Platzierung der Sätze wesentlich reduzieren (siehe Seite 162). Wenn Sie von dieser Reihenfolge abweichen, muss UDS/SQL meist größere Teile der Set-Occurrence absuchen, was verhältnismäßig viele Plattenzugriffe bedeuten kann. Insbesondere muss UDS/SQL zum Lesen des Vorgängers eines Satzes die Set-Occurrence von Beginn an absuchen, wenn Sie nicht die Rückwärtsverkettung anlegen.

  Wenn sich die physische Lage der Membersätze häufig ändert (z.B. wenn die Membersätze für einen anderen Set als Liste gespeichert sind), sind gelegentliche Reorganisationsläufe notwendig, um die Probable Position Pointer (PPP) aktuell zu halten (siehe Handbuch „Sichern, Informieren und Reorganisieren“, BREORG).

- Direktzugriff

  – ORDER IS SORTED INDEXED:
    Die Vorgänge sind analog zum Direktzugriff bei Adressliste.

  – ORDER IS LAST/FIRST/NEXT/PRIOR/SORTED:
    UDS/SQL muss durchschnittlich die halbe Set-Occurrence absuchen, bis der Satz gefunden ist. Dabei ist maximal für jeden Satz ein Plattenzugriff nötig.

- Einfügen

  Beim Einfügen wird der Satz mit seinem Vorgänger und Nachfolger über Zeiger verkettet.
  Deshalb muss UDS/SQL den Vorgänger, bei Rückwärtsverkettung auch den Nachfolger des einzufügenden Satzes ändern.

  – ORDER IS SORTED [INDEXED]:
    Den Einfügungsort eines Satzes findet UDS/SQL durch einen Direktzugriff.

  – ORDER IS LAST/FIRST/NEXT:
    Vom Ownersatz verweist entweder ein Zeiger auf den Vorgänger, oder der Ownersatz ist selbst der Vorgänger oder der Vorgänger ist durch die Currency-Information unmittelbar bekannt. Der Satz kann durch maximal zwei Plattenzugriffe eingefügt werden.

  – ORDER IS PRIOR:
    UDS/SQL muss die Set-Occurrence vom Beginn an absuchen, wenn Sie nicht die Rückwärtsverkettung anlegen.
    Bei Rückwärtsverkettung lässt sich ein Satz mit maximal 3 Plattenzugriffen einfügen.

- Löschen

  Neben dem zu löschenden Satz muss UDS/SQL dessen Vorgänger aufsuchen. Dies können Sie durch zusätzliche Rückwärtsverkettung beschleunigen. Bei der Rückwärtsverkettung muss UDS/SQL jedoch auch den Zeiger im Nachfolger ändern.
  Bei ORDER IS SORTED INDEXED muss der zum Satz gehörige Eintrag in der Sort-Key-Tabelle gelöscht werden.

- Ergebnis

  MODE IS CHAIN führt zu schnellen sequenziellen Zugriffen, wenn Sie die Sätze in der Reihenfolge verarbeiten, die durch die ORDER-Klausel entstanden ist. Ist bei der ORDER-Klausel SORTED INDEXED angegeben, sind auch bei großen Set-Occurrences schnelle Direktzugriffe möglich. Wurde ORDER IS SORTED angegeben, werden die Einfügungen langsamer, abhängig von der Anzahl der Membersätze in der Set-Occurrence. Änderungen des Primärschlüssels sind relativ aufwändig.

  Wenn die Reihenfolge der Membersätze unerheblich ist, wird die schnellste Satzspeicherung erreicht, wenn MODE IS CHAIN und ORDER IS NEXT bzw. ORDER IS FIRST angegeben wird.

5.3.3 Redundanz in SEARCH-Key-Tabellen verhindern

```
TYPE IS DATABASE-KEY-LIST
```

Sie haben bereits in der Beschreibung der logischen Datenstruktur mit der Schema-DDL darüber entschieden, ob UDS/SQL eine SEARCH-Key-Tabelle (SEARCH KEY IS... USING INDEX) anlegen soll.
Mit der TYPE-Klausel können Sie die Form einer SEARCH-Key-Tabelle beeinflussen.

Standardmäßig ist eine SEARCH-Key-Tabelle so aufgebaut wie eine mehrstufige Adressliste: Sie enthält für jeden Satz der zugehörigen Satzart bzw. jeden Membersatz der zugehörigen Set-Occurrence eine Zeile, die aus dem Schlüsselwert des Satzes und aus dem Zeiger zum Satz besteht.
(Diesem Standardfall entspricht die Angabe TYPE IS REPEATED-KEY.)

Häufig aber können verschiedene Sätze gleiche Schlüsselwerte besitzen (DUPLICATES ARE ALLOWED). Dann können Sie durch die Angabe TYPE IS DATABASE-KEY-LIST erreichen, dass der Schlüsselwert nicht mehrfach, sondern nur einmal stellvertretend für alle vorkommenden Duplikate abgespeichert wird.

Diese Art der SEARCH-Key-Tabelle heißt Duplikat-Tabelle. Duplikat-Tabellen sind insbesondere dann geeignet, wenn

– die Schlüssellänge groß ist,

– mehrere Schlüsselwerte mehr als 5 und weniger als 2000 Duplikate besitzen und

– die Verarbeitung größerer Satzmengen vorgesehen ist (siehe Handbuch „Anwendungen programmieren“, FIND 7).

In folgenden Fällen muss UDS/SQL eine dritte Überlaufseite anlegen:

– bei mehr als ca. 2000 Duplikaten in einer 2-Kbyte- oder 4-Kbyte-Datenbank

– bei mehr als ca. 4000 Duplikaten in einer 8-Kbyte-Datenbank

Hierbei sollten Sie den Vorteil der Platzersparnis gegen ungünstigeres Zugriffsverhalten (ein zusätzlicher Zugriff je Überlaufseite) abwägen.
Duplikat-Tabellen legt UDS/SQL nur auf der untersten Tabellenstufe an. Übergeordnete Tabellenstufen sind vergleichbar mit denen einer Standard-SEARCH-Key-Tabelle.

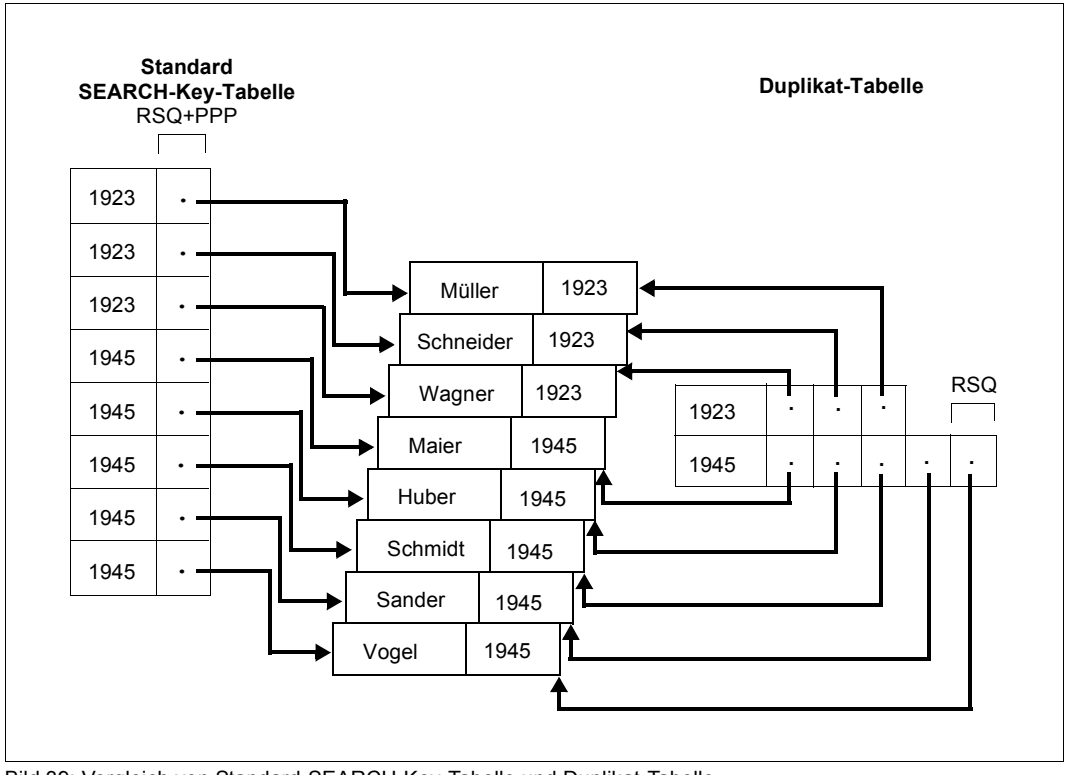


Bild 39: Vergleich von Standard-SEARCH-Key-Tabelle und Duplikat-Tabelle

Die Zeiger der Duplikat-Tabelle sind die RSQs der zugehörigen Sätze. Die zum gleichen Schlüsselwert gehörigen Zeiger sind aufsteigend nach RSQs sortiert.
Um die Seiten zu finden, in denen die Sätze gespeichert sind, ist daher ein zusätzlicher Zugriff auf die DBTT erforderlich.

5.3.4 Zusätzlichen Zeiger vom Member auf seinen Owner anlegen

```
MEMBER IS PHYSICALLY LINKED TO OWNER
```

Durch diese Angabe erreichen Sie, dass jeder Membersatz eines Set einen Zeiger auf den zugehörigen Ownersatz erhält. Der Zeiger ist ein Probable Position Pointer (PPP). Er optimiert den Zugriff auf einen Ownersatz, wenn einer seiner Membersätze bereits ausgewählt ist (siehe Handbuch „Anwendungen programmieren“, FIND 6). Solche Zugriffe treten z.B. häufig bei der Stücklistenverarbeitung auf.

Die Angabe ist für einen SYSTEM-Set verboten.

5.4 Lage von Membersätzen, Tabellen und Hashbereichen bestimmen

Die SSL bietet Möglichkeiten, die Lage folgender Objekte zu bestimmen:

- Membersätze
- Listen
- Adresslisten
- Sort-Key-Tabellen
- SEARCH-Key-Tabellen
- DBTTs
- Hashbereiche

Im Einzelnen haben Sie folgende Möglichkeiten:

- Bei Membersätzen, Listen, Adresslisten, Sort-Key-Tabellen und Set-SEARCH-KEY-Tabellen können Sie sowohl den Realm bestimmen, als auch innerhalb eines Realm die Konzentration dieser Daten auf eine Seite oder auf mehrere zusammenhängende Seiten erreichen.
- Bei Hashbereichen, DBTTs und Satz-SEARCH-Key-Tabellen können Sie nur den Realm bestimmen, in dem diese untergebracht werden sollen.

5.4.1 Lage von Membersätzen, zugehörigen Tabellen und Hashbereichen für Set-Sekundärschlüssel bestimmen

In diesem Abschnitt werden die Möglichkeiten zur Lagebestimmung von Daten gezeigt, die zu einer Set-Occurrence gehören. Diese Daten können sein:

- Ownersatz
- Membersätze bzw. Liste
- Adressliste
- Sort-Key-Tabelle
- Set-SEARCH-Key-Tabelle
- Hashbereich für Set-Sekundärschlüssel

5.4.1.1 Lagebestimmung auf Realm-Ebene

Mit folgenden Klauseln ordnen Sie Ihren Daten bestimmte Realms zu, ohne auf die Position innerhalb des Realm Einfluss zu nehmen:

Ownersatz, Membersätze

WITHIN-Klausel der Schema-DDL (siehe Abschnitt „Verteilung von Sätzen auf Realms bestimmen“ auf Seite 105)

Liste

```
MODE IS LIST DETACHED [WITHIN realmname]
```

Adressliste

```
MODE IS POINTER-ARRAY DETACHED [WITHIN realmname]
```

Sort-Key-Tabelle, Set-SEARCH-Key-Tabelle, Hashbereich für Set-Sekundärschlüssel

```
INDEX NAME IS name PLACING IS DETACHED [WITHIN realmname]
```

Mit *realmname* geben Sie den Realm an, in dem die Liste, Adressliste, Sort-Key-Tabelle, Set-SEARCH-Key-Tabelle bzw. der Hashbereich gespeichert werden soll.

Bei einer verteilbaren Liste bestimmt *realmname* den Tabellenrealm (siehe Seite 145) und den Realm, in dem ein evtl. genutzter indirekter Hashbereich der gesamten Liste gespeichert werden soll. Der Realm muß in der DDL-WITHIN-Klausel der Membersatzart genannt sein.
Falls die verteilbare Liste ohne DETACHED WITHIN-Klausel genutzt wird, ist die Lage der Tabellenseiten und eines indirekten Hashbereiches durch den erstgenannten Realm in der DDL-WITHIN-Klausel der Membersatzart bestimmt.

Wenn Sie keine Angabe machen, platziert UDS/SQL Listen, Adresslisten bzw. Sort-Key-Tabellen in den Realm des zugehörigen Ownersatzes, falls der Set kein SYSTEM-Set ist. Im nicht dynamischen SYSTEM-Set liegen diese Tabellen bzw. Hashbereiche standardmäßig im ersten Realm der DDL-WITHIN-Klausel der Membersatzart.
Einen dynamischen Set speichert UDS/SQL in den Temporären Realm.

Die Lage einer Liste wird ohne DETACHED WITHIN-Klausel durch die Lage des Owners bestimmt, falls der Set kein SYSTEM-Set ist und die Sätze in mehreren Realms liegen können. In diesem Fall müssen die Realms der Owner- und Membersatzart in der DDL-WITHIN-Klausel übereinstimmen. Wenn die Membersatzart nur in einem Realm liegen kann, wird die Lage der Liste ohne DDL-WITHIN-Klausel durch diesen Realm bestimmt.

Wenn Sie für eine Set-SEARCH-Key-Tabelle oder einen Hashbereich keine Angabe machen, wählt UDS/SQL den Realm nach folgendem Prinzip:

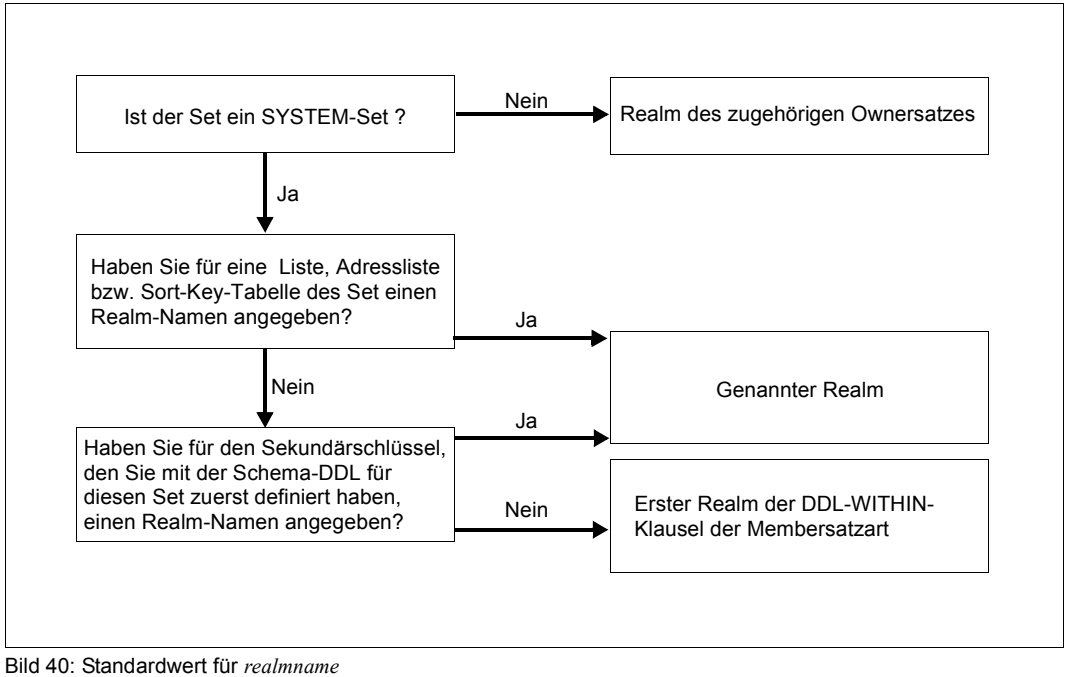


Bild 40: Standardwert für *realmname*

Mit *name* geben Sie den Namen der Sort- oder SEARCH-Key-Tabelle an, die Sie platzieren wollen. Den Namen müssen Sie mit der Schema-DDL vergeben haben (siehe Abschnitt „Hashbereiche und Tabellen benennen“ auf Seite 102).

5.4.1.2 Lagebestimmung innerhalb eines Realm

Innerhalb eines Realm können Sie die Daten, die zu einer Set-Occurrence gehören, auf einen zusammenhängenden Speicherplatz konzentrieren. Wenn Sie z.B. Membersätze als Liste speichern, werden die Membersätze physisch zusammenhängend in eine Seite gespeichert, bis die Liste die Seite ganz ausfüllt. Die anderen Möglichkeiten betreffen die Speicherung von Membersätzen und zugehörigen Tabellen in die Nähe des Ownersatzes. Die Möglichkeiten sind:

- Natürliche Optimierung:
  Listen, Adresslisten, Sort-Key-Tabellen, Set-SEARCH-Key-Tabellen und Membersätze in Ownernähe
- PLACEMENT OPTIMIZATION:
  Membersätze in Ownernähe
- MODE-Klausel:
  Adressliste bzw. Liste in die Ownerseite
- INDEX-Klausel:
  Sort-Key-Tabelle oder Set-SEARCH-Key-Tabelle in die Ownerseite

Natürliche Optimierung

Immer wenn Sie nicht durch DDL- und SSL-Angaben die Lage von Daten innerhalb eines Realm beeinflussen, speichert UDS/SQL die Daten in der physischen Reihenfolge, in der sie zeitlich anfallen. So können Sie beim Urladen der Datenbank oder später durch Entlade- und Ladeprogramme die Ladefolge z.B. so wählen, dass eine komplette Set-Occurrence mit allen zugehörigen Tabellen in aufeinander folgenden Seiten gespeichert wird. Anders als die übrigen Optimierungsmethoden wirkt die natürliche Optimierung über mehrere Hierarchiestufen. Deshalb ist diese Methode besonders geeignet, wenn Sie kritische Zugriffspfade erkennen. In diesem Fall speichern Sie die Daten in der Reihenfolge, die die Daten auf dem kritischen Zugriffspfad einnehmen.

Beispiel

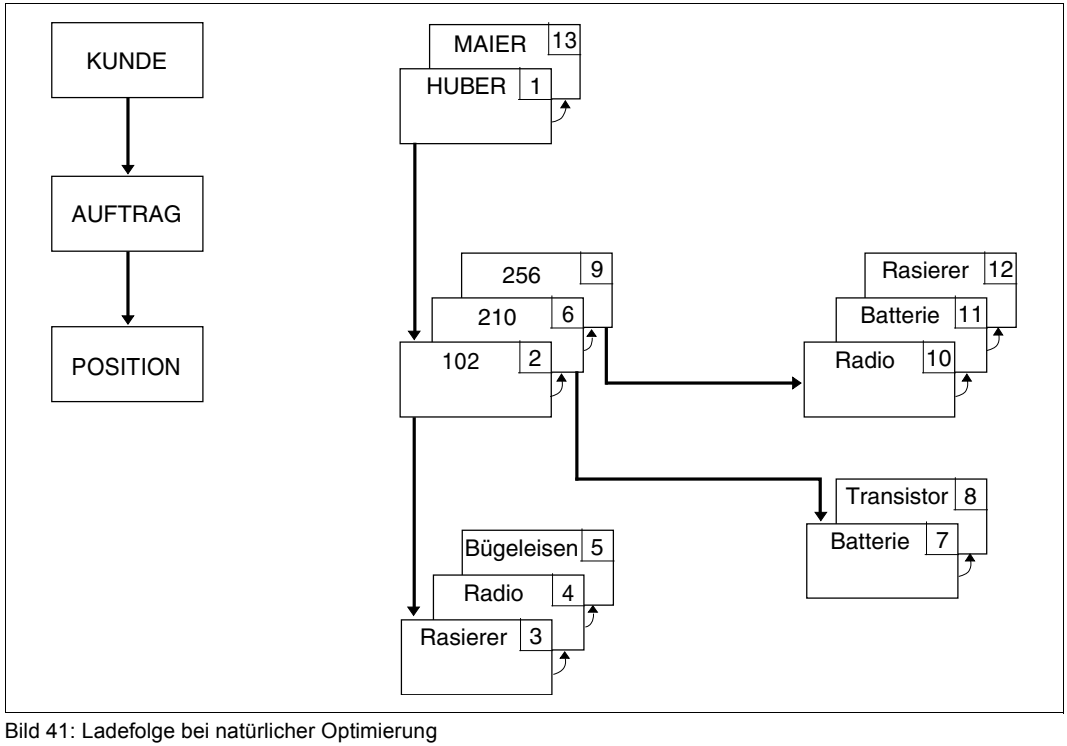


Bild 41: Ladefolge bei natürlicher Optimierung

In diesem Fall wird das Zugriffsverhalten auf dem kritischen Zugriffspfad Kunde, Auftrag, Position durch die angegebene Ladefolge verbessert. Das Schwergewicht wurde auf die Optimierung des Pfades Auftrag, Position gelegt.

PLACEMENT OPTIMIZATION

```
PLACEMENT OPTIMIZATION FOR SET setname
```

Mit *setname* nennen Sie den Namen des Set, den Sie als Zugriffspfad optimieren wollen. Er darf keinen SYSTEM-Set bezeichnen. Sie machen diese Angabe bei der Membersatzart dieses Set.
Sie erreichen damit, dass UDS/SQL beim Speichern eines Ownersatzes auf den Ownersatz unmittelbar folgend so viel Platz reserviert, dass er folgende Daten aufnehmen kann:

– Die Anzahl von Membersätzen, die Sie beim Mengengerüst festgelegt haben (siehe Abschnitt „Speicherplatzbedarf für Membersätze“ auf Seite 137).

– Falls die Membersätze Owner in anderen Sets sind:
Alle Tabellen, die Sie mittels ATTACHED in den Seiten dieser Sätze unterbringen wollen (siehe Abschnitt „MODE-Klausel“ auf Seite 163 und „INDEX-Klausel“ auf Seite 164).

Voraussetzung ist, dass Sie für die Ownersatzart des angegebenen Sets keine Lagebestimmung mit

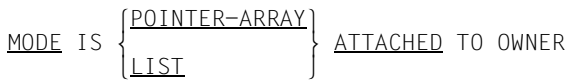
– LOCATION MODE IS CALC
– MODE IS LIST oder
– PLACEMENT OPTIMIZATION

vorgenommen haben.

Bei einer Satzart, die Sie in der Schema-DDL mit LOCATION MODE IS CALC definiert haben, erzeugt PLACEMENT OPTIMIZATION indirekte CALC-Seiten.

Wird der Set mit MODE IS LIST definiert, so wird PLACEMENT OPTIMIZATION ignoriert.

MODE-Klausel



Diese Angabe ist in SYSTEM-Sets nicht erlaubt.

Bei MODE IS LIST ist diese Angabe nicht erlaubt, wenn die Ownersatzart Membersatzart einer verteilbaren Liste ist.

Wenn Sie beim Mengengerüst eine Set-Occurrence-Population für den Set größer als 0 angegeben haben (siehe Abschnitt „Größe der Set-Occurrences eines Set angeben“ auf Seite 136), richtet UDS/SQL alle Tabellen einer Set-Occurrence beim Abspeichern des Ownersatzes und auf den Ownersatz folgend ein.
Eine Tabelle, die Sie ATTACHED speichern, bevorzugt UDS/SQL vor den anderen Tabellen für den Speicherplatz, der unmittelbar auf den Owner folgt.
Falls der Owner in einem Set Member ist, für den Sie über PLACEMENT OPTIMIZATION Speicherplatz reservieren, so berücksichtigt UDS/SQL auch den Speicherplatzbedarf einer zugehörigen ATTACHED-Tabelle, damit der Ownersatz mit der Tabelle zusammenhängend gespeichert werden kann (siehe Abschnitt „PLACEMENT OPTIMIZATION“ auf Seite 162).

Wenn Sie die Set-Occurrence-Population mit 0 angegeben haben, legt UDS/SQL eine Tabelle erst beim Einspeichern des ersten Membersatzes an. ATTACHED bewirkt dann, dass UDS/SQL die Tabelle in größtmöglicher Nähe zum Ownersatz anlegt.

Mit ATTACHED erreichen Sie also, dass ein Ownersatz und die zugehörige Tabelle durch nur einen Plattenzugriff verfügbar sind, wenn in der Ownerseite genügend Platz für die Tabelle ist.

Voraussetzung ist, dass Sie die Ownersatzart nicht mit LOCATION MODE IS CALC definiert haben, da es innerhalb von Hashbereichen keine Platzreservierung für Membersätze und Tabellen gibt.

INDEX-Klausel

```
INDEX NAME IS name PLACING IS ATTACHED TO OWNER
```

Diese Angabe ist in SYSTEM-Sets nicht erlaubt. Sie platzieren damit Tabellen von Primär- und Sekundärschlüsseln in die Nähe des zugehörigen Owners.

Die Wirkung von ATTACHED ist genauer bei der MODE-Klausel beschrieben.

Mit *name*, geben Sie den Namen der zu platzierenden Tabelle an. Den Namen müssen Sie mit der Schema-DDL vergeben haben (siehe Abschnitt „Hashbereiche und Tabellen benennen“ auf Seite 102).

5.4.2 Lage von Satz-SEARCH-Key-Tabelle, DBTT und Satz-Hashbereichen bestimmen

Bei diesen Daten können Sie nur den Realm bestimmen, in dem sie untergebracht werden sollen.

Mit folgenden Klauseln ordnen Sie Ihren Daten bestimmte Realms zu:

Satz-SEARCH-Key-Tabelle

```
INDEX NAME IS name PLACING IS WITHIN realmname
```

DBTT

```
DATABASE-KEY-TRANSLATION-TABLE WITHIN realmname
```

Hashbereich für Primärschlüssel

```
POPULATION IS {ganzzahl WITHIN realmname},...
```

Hashbereich für Satz-Sekundärschlüssel

```
INDEX NAME IS name PLACING IS WITHIN realmname
```

Mit *name* geben Sie den Namen der Tabelle bzw. des Hashbereichs an, den Sie platzieren wollen. Den Namen müssen Sie mit der Schema-DDL vergeben haben (siehe Abschnitt „Hashbereiche und Tabellen benennen“ auf Seite 102).

Mit *realmname* geben Sie den Realm an, in dem die Tabelle bzw. der Hashbereich gespeichert werden soll. Wenn Sie keinen Realm-Namen angeben, nimmt UDS/SQL für die DBTT und die Sekundärschlüssel standardmäßig den ersten Realm der DDL-WITHIN-Klausel dieser Satzart an.
Der Hashbereich für Primärschlüssel wird immer abhängig von der DDL-WITHIN-Klausel angelegt.

Beispiel

```
DDL: SET NAME IS ERTEILTE-AUFTRAEGE
            .
            .
            .
         OWNER IS KUNDE.
       MEMBER IS AUFTRAG
            .
            .
            .

SSL: RECORD NAME IS AUFTRAG
         PLACEMENT OPTIMIZATION FOR SET ERTEILTE-AUFTRAEGE.

     SET NAME IS ERTEILTE-AUFTRAEGE
         POPULATION IS 10
         MODE IS POINTER-ARRAY ATTACHED TO OWNER
         INDEX NAME IS SEARCH-TAB-ERT-AUFTR
             PLACING IS DETACHED.
```

Bei diesen Angaben ordnet UDS/SQL die Sätze und Tabellen des Set folgendermaßen an:

Bild 42: Anordnung der Daten zu vorstehender Definition

5.4.3 Übersicht über lagebestimmende Anweisungen

Lagebestimmende Anweisungen für Sätze:

Art der Daten	WITHIN realm-name,...	Membersätze im Set mit				LOCA-TION MODE IS CALC	Lage der Sätze
		OWNER IS SYSTEM	PLACEMENT OPTIMIZA-TION FOR SET	MODE IS ...			
Sätze	obligato-risch an-zugeben	Ja	Ja	unzulässig			
			Nein	LIST	siehe Tabelle 9 auf Seite 168		
				POINTER-ARRAY	ATTACHED	unzulässig	
					DETACHED	Ja	Hashbereich im WITHIN-Realm
						Nein	im WITHIN-Realm
					DETACHED WITHIN	Ja	Hashbereich im WITHIN-Realm
						Nein	im WITHIN-Realm
				CHAIN	-	Ja	Hashbereich im WITHIN-Realm
						Nein	im WITHIN-Realm
		Nein	Ja	LIST	ATTACHED	-	Liste beim Owner
					DETACHED	-	Liste beim Owner
					DETACHED WITHIN	unzulässig	
				POINTER-ARRAY/ CHAIN	-	Ja	Hashbereich (indirekter CALC) im WITHIN-Realm; Sätze beim Owner
						Nein	beim Owner
			Nein	LIST	siehe Tabelle 10 auf Seite 168		
				POINTER-ARRAY/ CHAIN	-	Ja	Hashbereich in WITHIN-Realm
						Nein	im WITHIN-Realm

Tabelle 8: Lagebestimmende Anweisungen für Sätze

Lagebestimmung für singuläre Listen:

MODE IS ...			**Lage der Sätze**
LIST	ATTACHED	-	unzulässig
	DETACHED	nicht verteilbare Liste	erster Realm in DDL-WITHIN-Klausel der Membersatzart
		verteilbare Liste	Realms der DDL-WITHIN-Klausel der Membersatzart
	DETACHED WITHIN	-	Liste im DETACHED-WITHIN-Realm

Tabelle 9: Lagebestimmung für singuläre Listen

Lagebestimmung für reguläre Listen ohne Placement Optimization:

MODE IS ...		**Owner**	**Memberrealm**	**Lage der Sätze**
LIST	ATTACHED	Owner in verteilbarer Liste	-	unzulässig
		Owner nicht in verteilbarer Liste	-	Liste beim Owner [1)]
	DETACHED	Owner in verteilbarer Liste	1 Memberrealm	Liste im Memberrealm
			mehrere Memberrealms	unzulässig
		Owner nicht in verteilbarer Liste	1 Memberrealm	Liste im Memberrealm
			mehrere Memberrealms	Liste beim Owner [1)]
	DETACHED WITHIN	-	-	Liste im DETACHED WITHIN-Realm [2]

Tabelle 10: Lagebestimmung für reguläre Listen ohne Placement Optimization

1 Die in der DDL-WITHIN-Klausel angegebenen Realms müssen für Owner- und Membersatzart gleich sein

2 Der DETACHED WITHIN-Realm muß in der DDL-WITHIN-Klausel der Membersatzart vorhanden sein.

Lagebestimmende Anweisungen für Tabellen und Hashbereiche:

Art der Daten	**Settyp**	**Angaben aus MODE-, INDEX- bzw. DBTT-Klausel**	**Lage**	
			Realm	**innerhalb des Realms**
Liste Adressliste Sort-Key-Tabelle Set-SEARCH-Key-Tabelle	nicht SYSTEM	ATTACHED TO OWNER	Ownerrealm	beim Owner
	SYSTEM	DETACHED [1)]	erster Realm in WITHIN-Klausel der Membersatzart	entsprechend Speicherzeitpunkt
	nicht SYSTEM oder SYSTEM	DETACHED WITHIN *realmname*	angegebener Realm	
Adressliste	dynamisch		Temporärer Realm	-
Hashbereich für verteilbare Liste oder Hashbereich für Set-SEARCH-Key	SYSTEM	DETACHED	erster Realm in WITHIN-Klausel der Membersatzart	
		DETACHED WITHIN *realmname*	angegebener Realm	
Satz-SEARCH-Key-Tabelle	-	WITHIN *realmname*	angegebener Realm	entsprechend Speicherzeitpunkt
		standardmäßig	erster Realm in WITHIN-Klausel	
Hashbereich für Satz-SEARCH-Key		WITHIN *realmname*	angegebener Realm	-
		standardmäßig	erster Realm in WITHIN-Klausel	
Hashbereich für Primärschlüssel	-	-	Realms aus WITHIN-Klausel	
DBTT		DBTT WITHIN *realmname*	angegebener Realm	
		standardmäßig	erster Realm in WITHIN-Klausel	

Tabelle 11: Lagebestimmende Anweisungen für Tabellen und Hashbereiche

1 Für Set-SEARCH-Key-Tabellen im SYSTEM-Set entnehmen Sie die Standardwerte für *realmname* dem Bild 40

5.5 Reorganisationsaufwand für Tabellen festlegen

```
DYNAMIC REORGANIZATION SPANS ganzzahl PAGES
```

UDS/SQL führt automatisch Tabellenerweiterungen durch, wenn beim Einspeichern von Sätzen Bedarf entsteht, der über die im Mengengerüst veranschlagten Anfangsgrößen hinausgeht. Eine Tabelle, die noch nicht die Größe einer Seite hat, sich innerhalb der Seite jedoch nicht um mindestens zwei Tabellenzeilen ausdehnen kann, erweitert UDS/SQL durch Umspeichern in eine andere Seite. Erstreckt sich eine Tabelle jedoch über mehrere Seiten und belegt ein zu erweiternder Tabellenteil eine Seite vollständig, so kann UDS/SQL eine bestimmte Anzahl der Seiten nach freiem Platz durchsuchen, die die logisch benachbarten Tabellenteile enthalten.

Mit *ganzzahl* geben Sie die Anzahl der zu durchsuchenden Tabellenseiten an. Bei Adresslisten, Listen, Sort-Key-Tabellen und Set-SEARCH-Key-Tabellen ist die Angabe nur wirksam, wenn Sie den Set nicht mit ORDER IS LAST/FIRST definiert haben.

Die zu durchsuchenden Seiten sind gleichzeitig die Seiten, die Sie der dynamischen Reorganisation von UDS/SQL unterstellen:

Findet UDS/SQL innerhalb dieser Seiten freien Platz für einen neuen Tabelleneintrag, so verschiebt UDS/SQL diesen Platz an die Stelle, an der der Tabelleneintrag einzufügen ist.

Wenn jedoch alle durchsuchten Seiten vollkommen mit Tabelleneinträgen gefüllt sind, erweitert UDS/SQL die Tabelle um eine leere Seite hinter den durchsuchten Seiten. Die anschließenden Vorgänge hängen vom Einfügungsort des Tabelleneintrags ab:

1. Der Tabelleneintrag ist weder am Anfang noch am Ende der Tabelle einzufügen: UDS/SQL verteilt den Inhalt der durchsuchten Seiten zusammen mit dem neuen Tabelleneintrag gleichmäßig auf den um eine Seite vergrößerten Speicherbereich.
2. Der Tabelleneintrag ist am Ende der Tabelle anzufügen: Die neue Seite nimmt den letzten Tabelleneintrag und den neuen Tabelleneintrag auf.
3. Der Tabelleneintrag ist am Anfang der Tabelle anzufügen: Den Inhalt der ersten Tabellenseite speichert UDS/SQL bis auf den ersten Tabelleneintrag in die neue Seite. Die erste Seite nimmt den einzufügenden Eintrag auf.

Durch diese Vorgänge wird ein hoher Füllgrad der Tabellenseiten erzielt. In den Fällen 2) und 3) berechnet sich der Füllgrad nach der Formel

$$\text{Füllgrad [\%]} = \frac{n-1}{n} \times 100$$

wobei n die maximale Anzahl der Tabelleneinträge pro Seite bedeutet.

Im Fall 1 ist der Füllgrad abhängig von Ihrer Wahl von *ganzzahl*. Er kann folgenden Wert nicht unterschreiten:

$$\text{Mindestfüllgrad [\%]} = \frac{ganzzahl}{ganzzahl+1} \times 100.$$

Das heißt, dass Sie hohe Füllgrade am leichtesten erzielen, wenn Sie Sätze in einer sortierten Reihenfolge einspeichern. Hohe Füllgrade bedeuten weniger Speicherplatzbedarf für Tabellen und damit kürzere Zugriffswege.

Wenn Sie hohe Füllgrade durch dynamische Reorganisation erreichen wollen, müssen Sie beachten:

Hohe Werte für *ganzzahl* sind

– bei Adresslisten, Sort- und SEARCH-Key-Tabellen eher gerechtfertigt als bei Listen.
– bei statischen Satzbeständen eher gerechtfertigt als bei häufigen Änderungen.

Die dynamische Reorganisation wird nicht für dynamische Sets und bei mehrstufigen Tabellen nur auf der untersten Tabellenstufe durchgeführt. Duplikat-Tabellen können nicht dynamisch reorganisiert werden.

Der Standardwert für *ganzzahl* ist 2.

Beispiele

DYNAMIC REORGANIZATION SPANS 1 PAGES

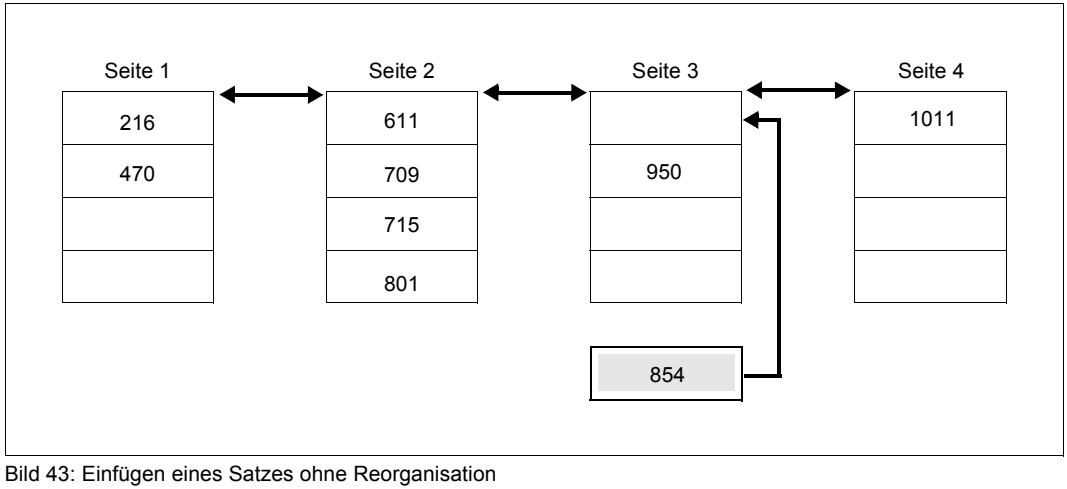


Bild 43: Einfügen eines Satzes ohne Reorganisation

Der Satz 854 kann ohne Reorganisation eingefügt werden.

Zum anschließenden Einfügen des Satzes 650 muss UDS/SQL eine neue Seite anlegen, da die Seite 2 ganz belegt ist. Die neue Seite nimmt so viel Einträge aus Seite 2 auf, dass eine gleichmäßige Verteilung in beiden Seiten besteht.

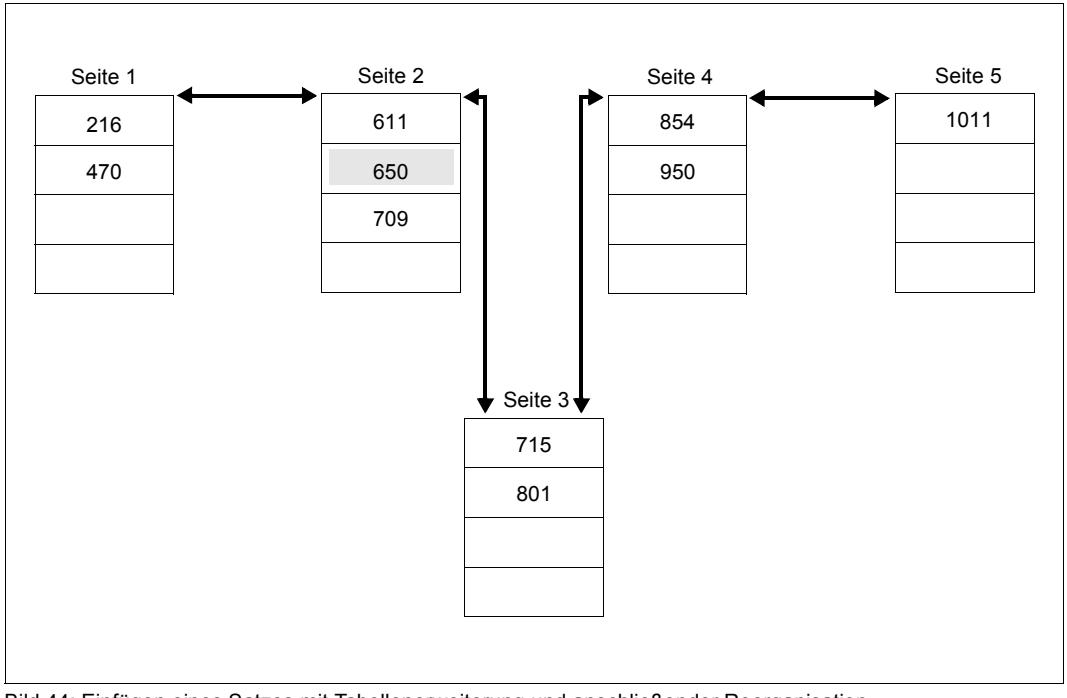


Bild 44: Einfügen eines Satzes mit Tabellenerweiterung und anschließender Reorganisation

DYNAMIC REORGANIZATION SPANS 3 PAGES

Aus der im Bild 43 gezeigten Ausgangslage entsteht durch Einfügen des Satzes 650 folgende Konstellation.

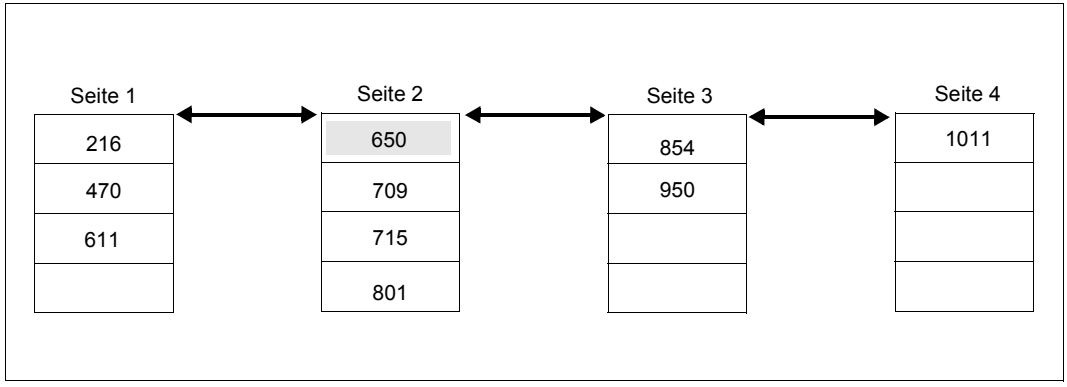


Bild 45: Einfügen eines Satzes durch Verschieben des Freiplatzes

5.6 Die Sätze einer Satzart komprimiert speichern

```
COMPRESSION FOR ALL ITEMS
```

Eine Satzart, die ein Feld variabler Länge enthält, dürfen Sie nicht komprimieren.
Mit der CALL-DML können Sie Sätze komprimiert speichern, indem Sie nur einen Teil der Felder speichern, die zu der Satzart gehören (siehe Handbuch „Anwendungen programmieren“, STORE 2).
Die COMPRESSION-Klausel benutzen Sie, wenn UDS/SQL die weggelassenen Felder nicht mit Standardwerten auffüllen soll, sondern die Sätze in der komprimierten Form speichern soll, wie Sie an der CALL-Schnittstelle vorliegen.

Bei einer Satzart, die Sie in der Schema-DDL mit LOCATION MODE IS CALC definiert haben, erzeugt die komprimierte Speicherung indirekte CALC-Seiten.

Wenn Sie diese Klausel angeben, können Sie mit SQL keine Sätze der Satzart ändern.

5.7 Berechnungsformeln für den Speicherplatzbedarf von Sätzen und Tabellen

Der Speicherplatzbedarf von Sätzen hängt davon ab, ob die Sätze in einen direkten Hashbereich gespeichert werden, ob für sie ein indirekter Hashbereich angelegt wird und welche Verknüpfung Sie für die Sätze bestimmt haben. Tabelle 12 und Tabelle 13 enthalten Formeln zur Berechnung des Speicherplatzbedarfs. Die Formeln unterscheiden sich, je nachdem, ob die Seitenlänge der Datenbank 2048 byte, 4000 byte oder 8096 byte beträgt.

Berechnungsformeln für eine Datenbank mit 2048 byte Seitenlänge

Anzahl der Sätze	in der Datenseite	$\frac{2028}{\text{Satzlänge}^{1}+8}$
	in der direkten CALC-Seite	$\frac{2018}{\text{Satzlänge}+\text{Schlüssellänge}+15}$
Anzahl der Einträge in der indirekten CALC-Seite		$\frac{2018}{\text{Schlüssellänge}+7}$
Anzahl der Tabellen-einträge pro Seite in einer	Liste	$\frac{2002}{\text{Satzlänge}+8}$
	Adressliste [2] Sort-Key-Tabelle [3] SEARCH-Key-Tabelle (TYPE IS REPEATED-KEY)	$\frac{2002}{\text{Schlüssellänge}+7}$

Tabelle 12: Berechnungsformeln für eine Datenbank mit 2048 byte Seitenlänge

Berechnungsformeln für eine Datenbank mit 4000 byte oder 8096 byte Seitenlänge

Anzahl der Sätze	in der Datenseite	$\frac{\text{Seitenlänge}-20}{\text{Satzlänge}^{1}+12}$
	in der direkten CALC-Seite	$\frac{\text{Seitenlänge}-30}{\text{Satzlänge}+\text{Schlüssellänge}+22}$
Anzahl der Einträge in der indirekten CALC-Seite		$\frac{\text{Seitenlänge}-30}{\text{Schlüssellänge}+10}$
Anzahl der Tabellen-einträge pro Seite in einer	Liste	$\frac{\text{Seitenlänge}-50}{\text{Satzlänge}+12}$
	Adressliste [2] Sort-Key-Tabelle [3] SEARCH-Key-Tabelle (TYPE IS REPEATED-KEY)	$\frac{\text{Seitenlänge}-50}{\text{Schlüssellänge}+10}$

Tabelle 13: Berechnungsformeln für eine Datenbank mit 4000 byte oder 8096 byte Seitenlänge

1
Satzlänge bedeutet Länge eines Satzes gemäß Schema-DDL zuzüglich der Zeigerlängen (siehe Seite 221, SCD)

2

```
                 ⎧LAST                              ⎫
                 ⎪FIRST                             ⎪
                 ⎪NEXT                              ⎪
Bei ORDER IS     ⎨                                  ⎬ gilt: Schlüssellänge = 0
                 ⎪PRIOR                             ⎪
                 ⎪IMMATERIAL                        ⎪
                 ⎩SORTED INDEXED BY DATABASE-KEY    ⎭
```

3

```
Bei ORDER IS SORTED INDEXED BY DATABASE-KEY gilt: Schlüssellänge = 0
```

5.8 Gesamtbeispiel für SSL

```
STORAGE STRUCTURE OF SCHEMA ARTIKELVERSAND.
*
*
RECORD NAME IS                    KUNDE
    DATABASE-KEY-TRANSLATION-TABLE IS 100.
*
RECORD NAME IS                    AUFTRAG
    DATABASE-KEY-TRANSLATION-TABLE IS 400
    PLACEMENT OPTIMIZATION FOR SET ERTEILTE-AUFTRAEGE.
*
RECORD NAME IS                    AUFTR-POS
    DATABASE-KEY-TRANSLATION-TABLE IS 1000.
*
RECORD NAME IS                    RATENZAHLUNG
    DATABASE-KEY-TRANSLATION-TABLE IS 50
    INDEX NAME IS SEARCH-TAB-RATENZAHLUNG
    TYPE IS DATABASE-KEY-LIST.
*
RECORD NAME IS                    ARTIKELART
    DATABASE-KEY-TRANSLATION-TABLE WITHIN ARTIKELRLM
    INDEX NAME IS SEARCH-TAB-ARTIKELART
    PLACING IS WITHIN ARTIKELRLM.
*
RECORD NAME IS                    ARTIKELAUSWAHL
    DATABASE-KEY-TRANSLATION-TABLE WITHIN ARTIKELRLM
    INDEX NAME IS SEARCH-TAB-ARTIKELAUSWAHL
    PLACING IS WITHIN ARTIKELRLM
    DYNAMIC REORGANIZATION SPANS 5 PAGES.
*
RECORD NAME IS                    ARTIKELBESCHR
    DATABASE-KEY-TRANSLATION-TABLE IS 300 WITHIN ARTIKELRLM
    POPULATION IS    200 WITHIN KLEIDUNG,
    100 WITHIN HAUSHALT,
    200 WITHIN SPORT,
    70 WITHIN LEBENSMITTEL,
    200 WITHIN SPIELE-HOBBY,
    100 WITHIN SCHREIBWAREN.
*
RECORD NAME IS                    ARTIKEL
    DATABASE-KEY-TRANSLATION-TABLE IS 600 WITHIN ARTIKELRLM
    POPULATION IS    400 WITHIN KLEIDUNG,
    100 WITHIN HAUSHALT,
    400 WITHIN SPORT,
    150 WITHIN LEBENSMITTEL,
    400 WITHIN SPIELE-HOBBY,
```

```
    250 WITHIN SCHREIBWAREN
    INDEX NAME IS SEARCH-TAB-ARTIKEL-1
    PLACING IS WITHIN ARTIKELRLM
    INDEX NAME IS SEARCH-TAB-ARTIKEL-2
    PLACING IS WITHIN ARTIKELRLM.
*
RECORD NAME IS                    MATERIALIEN
    INDEX NAME IS SEARCH-TAB-MATERIAL-1
    DYNAMIC REORGANIZATION SPANS 5 PAGES
    INDEX NAME IS SEARCH-TAB-MATERIAL-2
    DYNAMIC REORGANIZATION SPANS 5 PAGES.
*
RECORD NAME IS                    LIEFERANT
    DATABASE-KEY-TRANSLATION-TABLE IS 500
    POPULATION IS 200 WITHIN BESTELLRLM.
*
RECORD NAME IS                    BESTELLUNG
    DATABASE-KEY-TRANSLATION-TABLE IS 200.
*
RECORD NAME IS                    BESTELL-POS
    DATABASE-KEY-TRANSLATION-TABLE IS 500.
*
*
SET NAME IS                      ERTEILTE-AUFTRAEGE
    MODE IS POINTER-ARRAY ATTACHED TO OWNER
    POPULATION IS 10
    INDEX NAME IS SEARCH-TAB-ERT-AUFTR
    PLACING IS DETACHED.
*
SET NAME IS                      ANGEBOT
    MODE IS POINTER-ARRAY DETACHED WITHIN ARTIKELRLM
    WITH PHYSICAL LINK
    POPULATION IS 100 INCREASE IS 5
    DYNAMIC REORGANIZATION SPANS 5 PAGES.
*
SET NAME IS                      NAEHERE-AUSWAHL
    MODE IS POINTER-ARRAY DETACHED WITHIN ARTIKELRLM
    POPULATION IS 100 INCREASE IS 20
    DYNAMIC REORGANIZATION SPANS 5 PAGES.
*
SET NAME IS                      BESTELLANGABEN
    MODE IS LIST DETACHED WITH PHYSICAL LINK
    POPULATION IS 15
    MEMBER IS PHYSICALLY LINKED TO OWNER.
*
SET NAME IS                     MIN-BESTAND-ERREICHT
    MODE IS CHAIN LINKED TO PRIOR.
*
```

```
SET NAME IS                        ENTHAELT
    POPULATION IS 10
    MEMBER IS PHYSICALLY LINKED TO OWNER.
*
SET NAME IS                        ENTHALTEN-IN
    MODE IS CHAIN LINKED TO PRIOR
    MEMBER IS PHYSICALLY LINKED TO OWNER.
*
SET NAME IS                        LIEFERBARE-ARTIKEL
    POPULATION IS 500
    DYNAMIC REORGANIZATION SPANS 5 PAGES
    INDEX NAME IS SEARCH-TAB-LIEF-ART
    PLACING IS DETACHED
    TYPE IS DATABASE-KEY-LIST
    MEMBER IS PHYSICALLY LINKED TO OWNER.
*
SET NAME IS                        BESTELLTE-ARTIKEL
    MEMBER IS PHYSICALLY LINKED TO OWNER.
*
SET NAME IS                        NACHBESTELLTE-ARTIKEL
    MEMBER IS PHYSICALLY LINKED TO OWNER.
*
SET NAME IS                        BESTELL-INHALT
    MODE IS LIST DETACHED WITH PHYSICAL LINK
    POPULATION IS 20.
```

5.9 Reservierte Wörter des SSL-Compilers

ALL	AREA
ASSIGNED	ATTACHED
CHAIN	COMPRESSION
DATABASE-KEY-LIST	DATABASE-KEY-TRANSLATION-TABLE
DBKEY-TRANSLATION-TABLE	DBTT
DCB-NAME	DETACHED
DYNAMIC	FOR
INCREASE	INDEX
INDICATOR	IS
ITEMS	LINK
LINKED	LIST
MEMBER	MODE
NAME	OF
OPTIMIZATION	OWNER
PAGES	PHYSICAL
PHYSICALLY	PLACEMENT
PLACING	POINTER-ARRAY
POPULATION	PRIOR
RECORD	REORGANIZATION
REPEATED-KEY	SCHEMA
SET	SPANS
STORAGE	STRUCTURE
TO	TYPE
WITH	WITHIN

6 Definition der Benutzerschnittstelle der Datenbank

6.1 Subschema-DDL

6.1.1 Einführung

Im UDS/SQL-Schema müssen alle Daten definiert sein, die für die Aufgaben, die mit Hilfe der Datenbank gelöst werden sollen, notwendig sind. Das UDS/SQL-Schema hat jedoch keine direkte Schnittstelle zum Benutzer; Aspekte der benutzerfreundlichen Aufbereitung der Daten müssen beim Entwurf des UDS/SQL-Schemas nicht berücksichtigt werden.

Mit der Definition eines Subschemas schaffen Sie die Benutzerschnittstelle. Mit dem Subschema definieren Sie einen Ausschnitt aus dem Schema, der an die Erfordernisse eines abgegrenzten Aufgabengebietes angepasst ist.

Das hat den Vorteil, dass der Datenbankprogrammierer nur den Teil der Datenbank zu kennen braucht, den er für sein Aufgabengebiet benötigt. Ferner ist das Subschema eine Einheit für den Datenschutz, da dem Datenbankbenutzer nicht der gesamte Datenbestand zugänglich gemacht werden muss.

Sie haben folgende Möglichkeiten, die Datenstruktur des Schemas in eine, dem jeweiligen Verwendungszweck angepasste Subschema-Datenstruktur zu ändern:

- Satzarten ausschließen
- Sets ausschließen
- Realms ausschließen
- Satzelemente innerhalb einer Satzart ausschließen
- Wiederholungsfaktor von Vektoren und Wiederholungsgruppen verkleinern
- Satzelemente zu neuen Datengruppen zusammenfassen
- Bedingungen definieren

In diesem Kapitel ist beschrieben, wie Sie diese Funktionen durch die Definition eines Subschemas realisieren.

Die verwendete Metasprache ist auf Seite 18 erklärt, die allgemeinen Syntaxregeln sind auf Seite 230 zusammengefasst.

6.1.2 Benennung und Schutz des Subschemas

```
SUB-SCHEMA NAME IS subschemaname OF SCHEMA NAME schemaname
[PRIVACY LOCK FOR COMPILE IS literal-1[ OR literal-2]]
```

Mit *subschemaname* geben Sie dem Subschema einen Namen. *subschemaname* muss innerhalb einer Datenbank-Konfiguration in den ersten 6 Zeichen eindeutig sein.

schemaname bezeichnet das Schema, aus dem Sie das Subschema entnehmen.

Die Benutzung eines Subschemas zum Zugriff auf die Daten der Datenbank ist abhängig von den Zugriffsrechten des jeweiligen Benutzers. Diesen Datenschutz können Sie mit der Vergabe von Kennwörtern vervollständigen:

Mit *literal-1* und *literal-2* vergeben Sie Kennwörter, die die unbefugte Übersetzung eines DML-Programms mit diesem Subschema verhindert. Die Übersetzung kann dann nur erfolgen, wenn mindestens ein Kennwort bekannt ist.

6.1.3 Schema zur Entnahme eines Subschemas aufschließen

```
PRIVACY KEY FOR COPY IS literal.
```

Falls das Schema durch Kennwörter vor dem unbefugten Erstellen eines Subschemas geschützt ist, müssen Sie sich als zugriffsberechtigt ausweisen:

Mit *literal* müssen Sie eines der Kennwörter angeben, das zum Schutz des Schemas vereinbart wurde (siehe Abschnitt „Benennung und Schutz des Schemas“ auf Seite 107).

6.1.4 Satzarten komplett vom Schema ins Subschema übernehmen

Format 1:

```
COPY ALL RECORDS.
```

Format 2:

```
COPY satzname,... .
```

Format 1 benutzen Sie, wenn Sie alle im Schema vorkommenden Satzarten komplett ins Subschema übernehmen wollen.

Format 2 benutzen Sie, wenn Sie nicht alle im Schema vorkommenden Satzarten ins Subschema übernehmen wollen.

Mit *satzname* nennen Sie die komplett zu übernehmenden Satzarten.
Sie müssen alle Satzarten komplett oder teilweise übernehmen, die in den Sets des Subschemas enthalten sind.

6.1.5 Eine Satzart teilweise vom Schema ins Subschema übernehmen

```
01  satzname.
    {stufennummer satzelementname[ PICTURE.....][ USAGE.....]
                                                 [ OCCURS.....].}...
```

Mit *satzname* geben Sie den Namen einer Satzart an, die Sie teilweise ins Subschema übernehmen. Dies kann keine Satzart sein, die ein alphanumerisches Feld variabler Lange enthält. Solche Satzarten können Sie nur komplett ins Subschema übernehmen (siehe oben).

satzelementname bezeichnet

- ein Satzelement der Satzart, das Sie ins Subschema übernehmen. Sie übernehmen es gemäß Seite 184, Seite 186 (Feld) bzw. gemäß Seite 187 (Vektor) bzw. gemäß Seite 188 (Wiederholungsgruppe).
- eine Datengruppe, die Sie gemäß Seite 189 definieren.

Alle Felder, die am Aufbau der Subschema-Satzart beteiligt sind, müssen Sie genau in der Reihenfolge angeben, die der Schema-Beschreibung entspricht.

Ein numerisches Feld, ein alphanumerisches Feld fester Länge oder ein nationales Feld übernehmen

```
stufennummer feldname PICTURE IS maskenzeichenkette

                          ⎡DISPLAY        ⎤
                          ⎢COMPUTATIONAL-3⎥
                [USAGE IS ⎨               ⎬].
                          ⎢COMPUTATIONAL  ⎥
                          ⎣NATIONAL       ⎦
```

Über Stufennummer bestimmen Sie, ob das Feld einer Datengruppe angehört:
Soll das Feld nicht zu einer Datengruppe gehören, müssen Sie die Stufennummer so wählen, dass kein Satzelement der Satzart eine kleinere Stufennummer besitzt. Sie darf nicht kleiner als 02 sein.

Genau dann, wenn das Feld bereits im Schema einer Wiederholungsgruppe angehört, müssen Sie es auch im Subschema zum Bestandteil derselben Datengruppe erklären. Sie wählen die Stufennummer gemäß Seite 188.

Wollen Sie das Feld in eine neu definierte Datengruppe aufnehmen, gehen Sie gemäß Seite 189 vor.

feldname ist der Name, den Sie dem Feld im Schema gegeben haben.

Die Feldbeschreibung mit PICTURE- und USAGE-Klausel entnehmen Sie der Tabelle 14. Dabei bedeuten *n*, *m* und *l* Ganzzahlen.

Feldtyp	Feldbeschreibung im Schema	Feldbeschreibung im Subschema
numerisch, ungepackt	PICTURE IS *maskenzeichenkette*	PICTURE IS *maskenzeichenkette* [USAGE IS DISPLAY] *maskenzeichenkette* identisch mit der vom Schema
alphanumerisch, feste Länge		
	TYPE IS CHARACTER *m*	PICTURE IS X(*m*) [USAGE IS DISPLAY]
national	PICTURE IS N(*m*)	PICTURE IS N(*m*) [USAGE IS NATIONAL]
numerisch, gepackt	TYPE IS FIXED REAL DECIMAL *n*,*m* mit *n* > *m*, *m* > 0	PICTURE IS S9(*l*)V9(*m*) USAGE IS COMPUTATIONAL-3 *l*:= *n* - *m*
	mit *n* > 0, *m* < 0	PICTURE IS S9(*n*)P(-*m*) USAGE IS COMPUTATIONAL-3
	mit *n* < *m*	PICTURE IS SP(*l*)9(*n*) USAGE IS COMPUTATIONAL-3 *l*:= *m* - *n*
	mit *n* = *m*	PICTURE IS SV9(*n*) USAGE IS COMPUTATIONAL-3
	mit *m* = 0	PICTURE IS S9(*n*) USAGE IS COMPUTATIONAL-3
binär	TYPE IS FIXED REAL BINARY 15	PICTURE IS S9(*l*) USAGE IS COMPUTATIONAL *l*: = 1,2,3,4
	BINARY 31	PICTURE IS S9(*l*) USAGE IS COMPUTATIONAL *l*: = 5,6,7,8,9

Tabelle 14: Feldbeschreibung durch PICTURE- und USAGE-Klausel

Wenn Sie zu USAGE keine Angabe machen und die Maskenzeichenkette **nicht** das Symbol N enthält, wird standardmäßig DISPLAY angenommen. Wenn die Maskenzeichenkette das Symbol N enthält, wird bei fehlender USAGE-Klausel NATIONAL angenommen.

Ein Database-Key-Feld übernehmen

```
stufennummer feldname USAGE IS { DATABASE-KEY      }.
                               { DATABASE-KEY-LONG }
```

Über *stufennummer* bestimmen Sie, ob das Feld einer Datengruppe angehört:
Soll das Feld nicht zu einer Datengruppe gehören, müssen Sie die Stufennummer so wählen, dass kein Satzelement der Satzart eine kleinere Stufennummer besitzt. Sie darf nicht kleiner als 02 sein.
Genau dann, wenn das Feld bereits im Schema einer Wiederholungsgruppe angehört, müssen Sie es auch im Subschema zum Bestandteil derselben Datengruppe erklären. Sie wählen die Stufennummer gemäß Seite 188.
Wollen Sie das Feld in eine neu definierte Datengruppe aufnehmen, gehen Sie gemäß Seite 189 vor.

feldname ist der Name, den Sie dem Database-Key-Feld im Schema gegeben haben.
Ein Feld, dass Sie mit USAGE IS DATABASE-KEY-LONG ins Subschema übernehmen, muss im Schema als DATABASE-KEY-LONG-Feld definiert sein.

Einen Vektor übernehmen und ggf. verkürzen

```
stufennummer vektorname PICTURE IS maskenzeichenkette
                                 ⎧DISPLAY           ⎫
                                 ⎪                  ⎪
                                 ⎪COMPUTATIONAL-3   ⎪
                [USAGE IS        ⎨COMPUTATIONAL     ⎬]
                                 ⎪                  ⎪
                                 ⎪NATIONAL          ⎪
                                 ⎪                  ⎪
                                 ⎪DATABASE-KEY      ⎪
                                 ⎪                  ⎪
                                 ⎩DATABASE-KEY-LONG ⎭
               [OCCURS ganzzahl TIMES].
```

Ein Vektor ist ein Feld mit Wiederholungsfaktor. Der Wiederholungsfaktor gibt an, wie viel Duplikate des Feldes zu dem Vektor zusammengefasst werden.

Die Übernahme eines Vektors ins Subschema definieren Sie genauso, wie die Übernahme eines Feldes (siehe Abschnitt „Ein numerisches Feld, ein alphanumerisches Feld fester Länge oder ein nationales Feld übernehmen“ auf Seite 184 und Abschnitt „Ein Database-Key-Feld übernehmen“ auf Seite 186).

Mit *ganzzahl* geben Sie zusätzlich den Wiederholungsfaktor an, der mindestens 1 betragen muss und höchstens so groß sein darf wie im Schema. Das heißt, Sie können einen Vektor auf beliebig viele Felder, im Extremfall bis auf ein Feld verkürzen, wenn Sie ihn ins Subschema übernehmen.

Standardwert für *ganzzahl* ist 1.

Eine Wiederholungsgruppe übernehmen und ggf. verkleinern

```
 stufennummer-1 datengruppenname[ GROUP-USAGE IS NATIONAL]
                                [ OCCURS ganzzahl TIMES].

{stufennummer-2 satzelementname PICTURE.....
                                USAGE.....
                                     OCCURS.....}...
```

Eine Wiederholungsgruppe ist eine Datengruppe mit Wiederholungsfaktor. Der Wiederholungsfaktor gibt an, wie viel Duplikate dieser Datengruppe zu der Wiederholungsgruppe zusammengefasst werden.

Sie müssen eine Wiederholungsgruppe ins Subschema übernehmen, wenn Sie eines ihrer Felder übernehmen wollen.

Sie können eine Wiederholungsgruppe bei der Übernahme ins Subschema jedoch wie folgt verkleinern:

– Sie geben einen geringeren Wiederholungsfaktor an. Im Extremfall existiert die Datengruppe dann nur in einfacher Ausfertigung.

– Sie übernehmen Satzelemente der Wiederholungsgruppe nicht ins Subschema.

datengruppenname bezeichnet den Namen, den Sie im Schema für die Wiederholungsgruppe vergeben haben.

Mit *ganzzahl* geben Sie den Wiederholungsfaktor an. *ganzzahl* muss mindestens 1 und darf höchstens so groß sein wie im Schema.
Standardwert für *ganzzahl* ist 1.

satzelementname bezeichnet ein Satzelement, das Sie im Schema zum Bestandteil der Wiederholungsgruppe erklärt haben und ins Subschema übernehmen wollen. Sie übernehmen es gemäß Seite 184, Seite 186 (Feld) bzw. gemäß Seite 187 (Vektor) bzw. gemäß dieses Abschnitts (Wiederholungsgruppe).

Über *stufennummer-2* müssen Sie das Satzelement auch im Subschema zum Bestandteil der Wiederholungsgruppe erklären. Dazu müssen Sie *stufennummer-2* größer wählen als *stufennummer-1*.

Für alle Satzelemente, die Sie aus der Wiederholungsgruppe des Schemas ins Subschema übernehmen, gilt außerdem: Satzelemente müssen die gleiche Stufennummer haben, wenn ihnen die nächsthöhere Datengruppe gemeinsam ist.

Mit der GROUP-USAGE-Klausel vereinbaren Sie eine nationale Wiederholungsgruppe, das ist eine Wiederholungsgruppe, die in ihrer Gesamtheit wie ein nationales Datenfeld behandelt wird. Die GROUP-USAGE-Klausel darf nur angegeben werden, wenn alle untergeordneten Satzelemente vom Typ NATIONAL sind und *stufennummer-1* ungleich 01 ist.

Satzelemente zu einer Datengruppe zusammenfassen

```
 stufennummer-1 datengruppenname[ GROUP-USAGE IS NATIONAL]
                                 [ OCCURS ganzzahl TIMES].

{stufennummer-2 satzelementname PICTURE.....
                                      USAGE.....
                                           OCCURS.....}...
```

Eine Datengruppe ist eine benennbare Zusammenfassung von Satzelementen innerhalb einer Satzart, wobei jedes Satzelement ein Feld, ein Vektor oder selbst eine Datengruppe sein kann.

Mit *datengruppenname* geben Sie der Datengruppe einen Namen.

satzelementname bezeichnet ein Satzelement, das Sie zum Bestandteil der Datengruppe erklären wollen. Es kann sein

– ein Satzelement, das Sie aus dem Schema ins Subschema übernehmen wollen. Sie übernehmen es gemäß Seite 184, Seite 186 (Feld) bzw. gemäß Seite 187 (Vektor) bzw. gemäß Seite 188 (Wiederholungsgruppe).

– eine Datengruppe, die Sie gemäß dieses Abschnitts definieren.

stufennummer-2 muss dabei größer sein als *stufennummer-1*.

Für alle Satzelemente, die Sie zur Datengruppe zusammenfassen, gilt außerdem: Satzelemente müssen die gleiche Stufennummer haben, wenn ihnen die nächsthöhere Datengruppe gemeinsam ist.

Mit der GROUP-USAGE-Klausel vereinbaren Sie eine nationale Datengruppe, das ist eine Datengruppe, die in ihrer Gesamtheit wie ein nationales Datenfeld behandelt wird. Die GROUP-USAGE-Klausel darf nur angegeben werden, wenn alle untergeordneten Satzelemente vom Typ NATIONAL sind und *stufennummer-1* ungleich 01 ist.

Eine Bedingung definieren

Detaillierte Informationen finden Sie im COBOL2000-Handbuch „Sprachbeschreibung".

```
88 bedingungsname

   {VALUE IS  }
   {          } {literal-1[ THROUGH literal-2]},... .
   {VALUES ARE}
```

Der Datenbankprogrammierer kann die Ausführung von Programmanweisungen von Bedingungen abhängig machen. Als Bedingung können Sie definieren, dass bestimmte Felder bestimmte Feldinhalte haben.
Die Felder werden dann als Bedingungsvariablen bezeichnet.

Mit *bedingungsname* geben Sie der Bedingung einen Namen, auf den sich der Datenbankprogrammierer beziehen kann. Die Namensvergabe muss unmittelbar der Beschreibung des Feldes folgen, das Bedingungsvariable sein soll.

Mit *literal-1*, *literal-2* usw. definieren Sie einen Wertebereich für die Bedingung. Die Bedingung ist dann erfüllt, wenn das Feld einen Wert aus diesem Wertebereich enthält. Der Wertebereich kann sich aus Einzelwerten und aus Intervallen von Werten zusammensetzen. Er muss in dem Wertebereich enthalten sein, den Sie für das Feld definiert haben.

Sie können für eine Bedingungsvariable mehrere Bedingungsnamen mit dem dazugehörigen Wertebereich definieren.

Beispiel

```
01  AUFTRAG
        .
        .
        .
    02   AUFTRAGS-STATUS PICTURE IS S9.
    88   ERLEDIGT VALUE IS 1.
    88   OFFEN    VALUE IS 0.
```

Die Bedingung ERLEDIGT ist erfüllt, wenn das Feld AUFTRAGS-STATUS den Wert 1 enthält. Wenn es den Wert 0 enthält, ist die Bedingung OFFEN erfüllt.

6.1.6 Sets vom Schema ins Subschema übernehmen

Format 1:

```
COPY ALL SETS.
```

Format 2:

```
COPY setname-1,... .
```

Format 1 benutzen Sie, wenn Sie alle im Schema vorkommenden Sets ins Subschema übernehmen wollen.

Format 2 benutzen Sie, wenn Sie nicht alle im Schema vorkommenden Sets ins Subschema übernehmen wollen.

Mit *setname* nennen Sie die zu übernehmenden Sets. Selbstverständlich müssen alle Owner- und Membersatzarten dieser Sets im Subschema enthalten sein.

Wenn Sie Schlüsselfelder ändern wollen, die sich auf einen Set beziehen, müssen die betroffenen Sets ebenfalls im Subschema vorhanden sein.

Wenn Sie auf eine Satzart mit SQL zugreifen, die in mehreren Sets Membersatzart ist, müssen zu dieser Satzart im Subschema auch alle Sets vorhanden sein.

6.1.7 Realms vom Schema ins Subschema übernehmen

Format 1:

```
COPY ALL AREAS.
```

Format 2:

```
COPY realmname,... .
```

Format 1 benutzen Sie, wenn Sie alle im Schema vorkommenden Realms ins Subschema übernehmen wollen.

Format 2 benutzen Sie, wenn Sie nicht alle im Schema vorkommenden Realms ins Subschema übernehmen wollen.

Mit *realmname* nennen Sie die zu übernehmenden Realms. Sie müssen alle Realms übernehmen, in denen Daten zu den Satzarten und Sets des Subschemas enthalten sind, die der Subschemabenutzer benötigt. Sie entnehmen der Tabelle, welche Realms das sein können. Wenn Sie mit SQL arbeiten, müssen Sie den Temporären Realm auf jeden Fall übernehmen.

Art der Daten	**Realm(s) aus**
Sätze	WITHIN-Klauseln (DDL)
DBTTs	DBTT-Klauseln
Hashbereiche für Primärschlüssel	Satz-POPULATION-Klauseln (SSL)
Hashbereiche für Satz-Sekundärschlüssel	Satz-INDEX-Klauseln (SSL)
Satz-SEARCH-KEY-Tabellen	
Adresslisten	MODE-Klauseln (SSL)
Listen	
Sort-Key-Tabellen Set-SEARCH-Key-Tabellen Hashbereiche für Set-Sekundärschlüssel	Set-INDEX-Klauseln (SSL)

Tabelle 15: Zu übernehmende Realms

6.1.8 Gesamtbeispiel für Subschema-DDL

```
IDENTIFICATION DIVISION.
    SUB-SCHEMA NAME IS ADMIN OF SCHEMA ARTIKELVERSAND
   PRIVACY KEY FOR COPY IS "VERSANDKEY".
*
*
*
DATA DIVISION.
*
AREA SECTION.
    COPY ALL AREAS.
*
RECORD SECTION.
    COPY ALL RECORDS.
*
SET SECTION.
    COPY ALL SETS.
*
*
*
```

6.2 Relationales Schema

Mit der Schema-DDL kann ein Schema nach relationalen Gesichtspunkten oder nach CODASYL-Gesichtspunkten erstellt werden. In einem Schema, das nach relationalen Regeln definiert ist, sind keine Set-Beziehungen enthalten, die Primär- und Fremdschlüssel sind vom Anwender definiert.

Wenn ein Schema nach CODASYL-Regeln erstellt wurde, dann liefert das Dienstprogramm BPSQLSIA dem SQL-Programmierer eine relationale Sicht auf das CODASYL-Subschema.

Das Dienstprogramm BPSQLSIA analysiert alle Objekte eines Subschemas und stellt sie als Objekte einer relationalen Datenbank dar.

Die Ergebnisliste der Umsetzung mit BPSQLSIA enthält alle Informationen, die für den Programmierer Voraussetzung sind, um mit SQL arbeiten zu können.

Damit ein CODASYL-Subschema vollständig relational mit SQL bearbeitet werden kann, muss es einige Bedingungen erfüllen. Einschränkungen und Hinweise zu DDL, SSL und Subschema-DDL finden Sie in den entsprechenden Kapiteln.

Das Dienstprogramm BPSQLSIA einschließlich einer Übersicht der Einschränkungen und einem Beispiel ist in einem anderen UDS/SQL-Handbuch beschrieben (siehe Handbuch UDS/SQL „Sichern, Informieren und Reorganisieren“, BPSQLSIA).

7 Aufbau der Seiten

Um eine Datenbank mit DDL und SSL aufbauen zu können, benötigen Sie im Allgemeinen keine Informationen über den Aufbau von Seiten, Sätzen und Tabellen.
Die beiden folgenden Kapitel dienen zum Nachschlagen, wenn Sie an speziellen Details interessiert sind.

Der gesamte Speicherplatz für die Daten der Datenbank ist in Realms aufgeteilt. Jeder Realm besteht aus Seiten, deren Anzahl beim Datenbankaufbau festgelegt wird und nachträglich geändert werden kann (siehe Handbuch „Aufbauen und Umstrukturieren“).

Die Länge einer Seite kann 2048 byte, 4000 byte oder 8096 byte betragen und ist einheitlich innerhalb einer Datenbank. Die Seitenlänge für eine Datenbank wird festgelegt beim Aufbauen der Datenbank mit dem Dienstprogramm BCREATE und kann nachträglich vergrößert werden mit dem Dienstprogramm BPGSIZE (siehe Handbuch „Aufbauen und Umstrukturieren“).

Seiten der Länge 4000 byte oder 8096 byte sind jeweils in einen Seitencontainer eingebettet: Vor der Seite liegt ein 64 byte langer Header, im Anschluss an die Seite liegt ein 32 byte langer Trailer.

Die folgenden Arten von Seiten sind am Realmaufbau beteiligt, diese Seiten unterscheiden sich durch ihren Dateninhalt und ihren Aufbau:

Act-Key-0-Seite
: enthält realmspezifische Daten; ist immer die 1. Seite im Realm.

Act-Key-N-Seite
: enthält realmspezifische Daten; ist immer die letzte Seite im Realm.

FPA-Seite
: Seite zur Freiplatzverwaltung auf Realm-Ebene. Mindestens eine FPA-Seite ist immer vorhanden (FPA: Free Place Administration).

DBTT-Ankerseite
: verwaltet DBTT-Bereiche einer Satzart (DBTT: Database Key Translation Table).

DBTT-Seite
: Seite eines DBTT-Bereichs. Verwaltet Sätze einer Satzart

CALC-Seite
: Seite eines Hashbereichs

Datenseite

enthält Sätze, die nicht in einen Hashbereich gespeichert werden und Tabellen (keine DBTT).

In den folgenden Abschnitten sind der Seitencontainer und die verschieden Arten von Seiten im Detail beschrieben. Die Beschreibung der Seiten beschränkt sich auf den Aufbau der Seiten selbst; Header und Trailer bei 4000 byte bzw. 8096 byte langen Seiten sind also nicht dargestellt. Die Distanzangaben zu den einzelnen Seitenbereichen beziehen sich jeweils auf den Seitenanfang. Wenn nichts anderes erwähnt ist, gilt die Beschreibung gleichermaßen für Seiten der Länge 2048 byte, 4000 byte und 8096 byte. Wo Unterschiede bestehen, wird gesondert darauf hingewiesen.

7.1 Seitencontainer

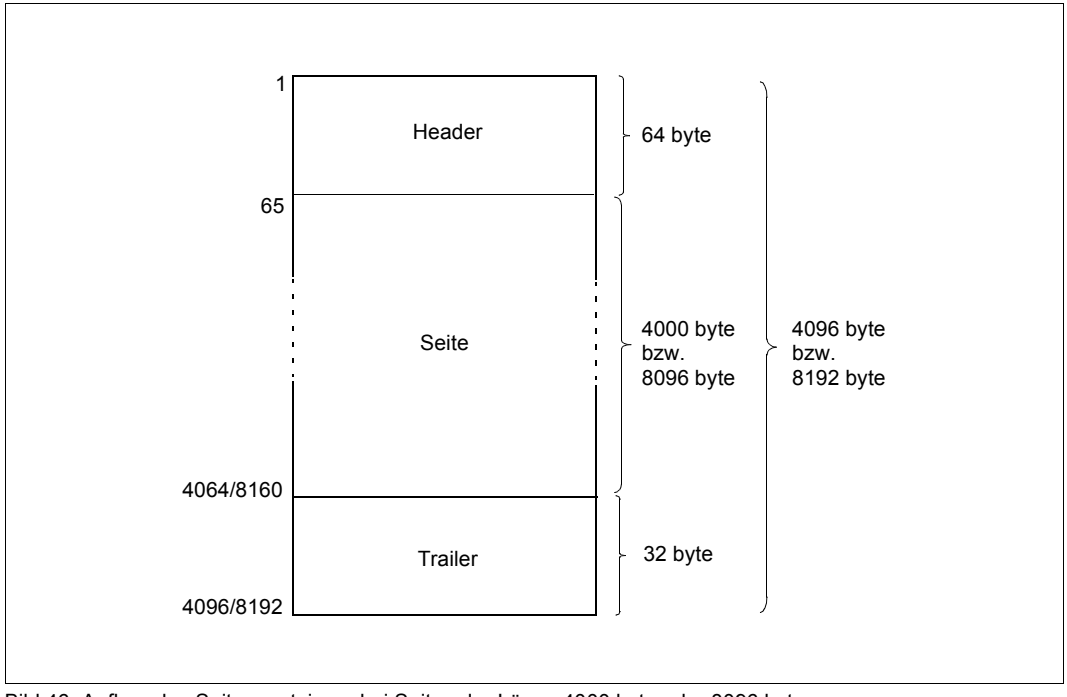


Bild 46: Aufbau des Seitencontainers bei Seiten der Länge 4000 byte oder 8096 byte

Aufschlüsselung der Bytes

Byte	**Bedeutung**
1-64	Header; die Bytes 17-24 enthalten eine Versionsnummer.
65-4064 bzw. 65-8160	Seite der Länge 4000 byte bzw. 8096 byte
4065-4096 bzw. 8161-8192	Trailer; die Bytes 4089-4096 (bei 4000-byte Seiten) bzw. die Bytes 8185-8192 (bei 8096-byte-Seiten) enthalten eine Versionsnummer.

Tabelle 16: Aufschlüsselung der Bytes des Seitencontainers

Header und Trailer werden von UDS/SQL zu Verwaltungszwecken genutzt. Bei einer konsistenten Seite stimmen die Versionsnummern in Header und Trailer überein.

7.2 Act-Key-0- und Act-Key-N-Seite

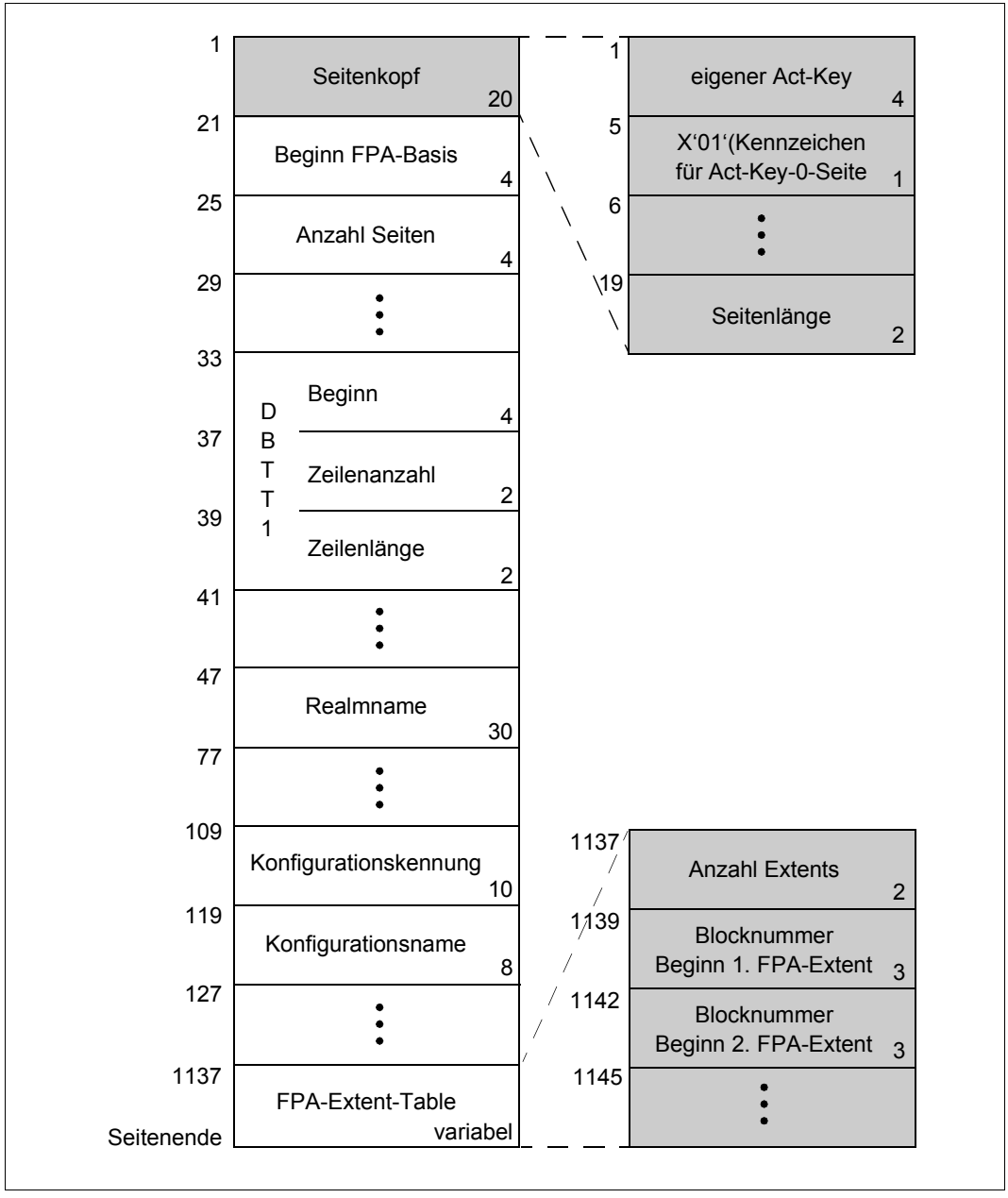


Bild 47: Aufbau der Act-Key-0- und der Act-Key-N-Seite

Aufschlüsselung der Bytes

Byte	Bedeutung
1-4	Der Act-Key gibt die Adresse der Seite an. Da die Act-Key-0-Seite immer die erste Seite eines Realm ist, enthält der Act-Key dieser Seite immer die Seitennummer 0.
19-20	Die Seitenlänge beträgt 2048 byte/4000 byte/8096 byte.
21-24	enthält den Act-Key der ersten FPA-Basis-Seite. Eine Beschreibung des Layouts einer FPA-Basis-Seite finden Sie im Abschnitt „FPA-Seite“ auf Seite 200.
25-28	gibt die Anzahl der Seiten des Realm an.
33-40	Diese Felder sind nur im Realm DBDIR[1] besetzt.
47-76	enthält den Realmnamen
109-126	Diese Felder sind nur im Realm DBDIR[1] besetzt.
ab 1137	enthält die FPA-Extent-Table. Eine Beschreibung des Layouts eines FPA-Extents und einer FPA-Extent-Seite finden Sie im Abschnitt „FPA-Seite“ auf Seite 200.

Tabelle 17: Aufschlüsselung der Bytes der Act-Key-0- und der Act-Key-N-Seite

[1] DBDIR ist das Database Directory

7.3 FPA-Seite

Die FPA-Seiten sind eine Stufe der insgesamt dreistufigen Freiplatzverwaltung in UDS/SQL. Während durch FPA-Seiten der freie Platz auf Realm-Ebene verwaltet wird, gibt es auch eine Freiplatzverwaltung auf Seiten- und auf Tabellenebene.

Falls sich in einer Datenbankseite Tabellen befinden, gibt der in einer FPA-Seite angegebene zugehörige Wert nicht immer den tatsächlich belegten Speicherplatz an.

Jede FPA-Seite enthält einen eigenen Act-Key, der die Adresse der Seite angibt. Das zu einem vorgegebenen Act-Key gehörende Paar (FPA-Seite, Eintrag innerhalb der Seite) kann mit Hilfe der Information über die FPA-Basis und die FPA-Extent-Table in der Act-Key-0-Seite gezielt bestimmt werden.

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, beträgt die Länge einer FPA-Seite 2048 byte, 4000 byte oder 8096 byte.

In einem Realm können neben der Basis-Freiplatzverwaltungstabelle (FPA-Basis) weitere Freiplatzverwaltungstabellen (FPA-Extents) vorhanden sein.

Die FPA-Basis besteht aus den FPA-Seiten, die beim Realm-Aufbau oder bei der Realm-Umstellung durch BPGSIZE angelegt werden. FPA-Extents können bei einer Realm-Erweiterung entstehen. Dabei ist es unerheblich, ob der Realm online erweitert wird oder im Offline-Modus durch das Dienstprogramm BREORG.

Jeder FPA-Extent ist 128KB groß. In 2KB-Datenbanken besteht er aus 64 Seiten, in 4KB-Datenbanken aus 32 Seiten und in 8KB-Datenbanken aus 16 Seiten. FPA-Extents werden über die Act-Key-0-Seite verwaltet. Mit dem Dienstprogramm BPRECORD können Sie sich Informationen zur FPA-Basis und den FPA-Extents ausgeben lassen (siehe Handbuch „Sichern, Informieren und Reorganisieren“).

Aufbau einer FPA-Basis-Seite der Länge 2048 byte

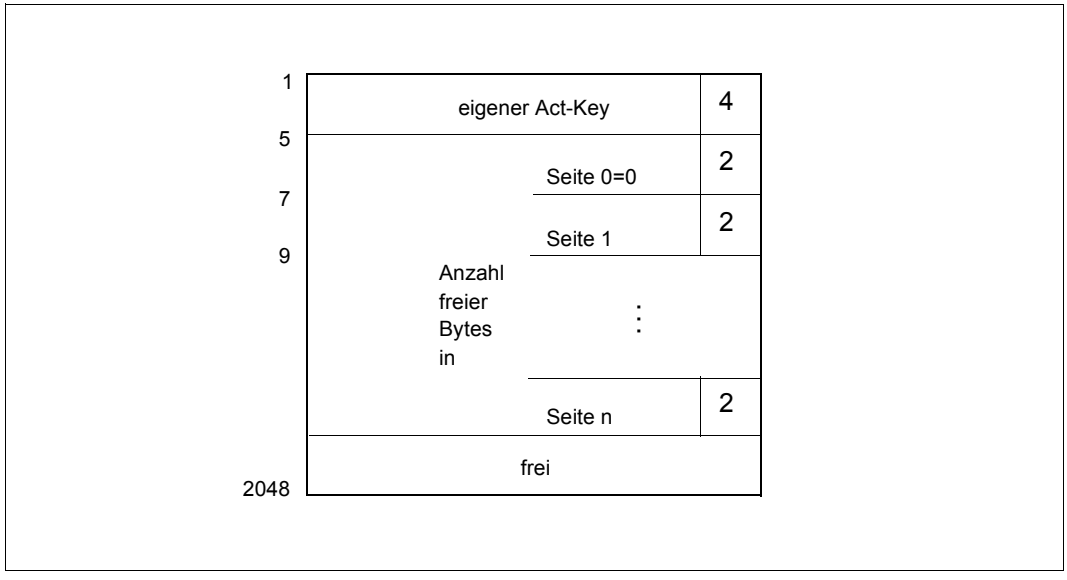


Bild 48: Aufbau einer FPA-Basis-Seite der Länge 2048 byte

Aufschlüsselung der Bytes

Byte	Bedeutung
1-4	Der Act-Key gibt die Adresse der Seite an.
5	Ab diesem Byte ist für jede Seite des Realm ein Eintrag von 2 byte Länge vorhanden, der die Anzahl freier Bytes in der jeweiligen Seite angibt. Sie beträgt 0 für – Act-Key-0-Seite, Act-Key-N-Seite – FPA-Seiten – DBTT-Ankerseiten, – DBTT-Seiten, – CALC-Seiten, da diese Seiten ganz für den jeweiligen Zweck reserviert sind und ihnen daher keine zusätzlichen Daten über die Freiplatzverwaltung zugewiesen werden können. Für leere Seiten ist der effektiv nutzbare Speicherplatz in einer Länge von 2028 byte angegeben, da der Seitenkopf 20 Bytes in Anspruch nimmt.

Tabelle 18: Aufschlüsselung der Bytes einer FPA-Basis-Seite der Länge 2048 byte

Aufbau einer FPA-Extent-Seite der Länge 2048 byte oder einer FPA-Seite der Länge 4000 byte oder 8096 byte

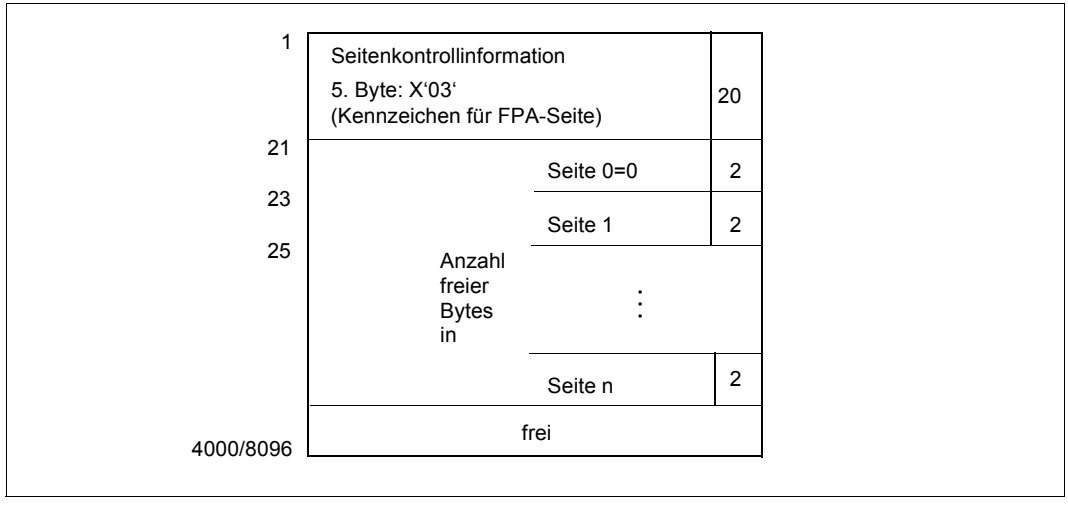


Bild 49: Aufbau einer FPA-Extent-Seite der Länge 2048 byte oder einer FPA-Seite der Länge 4000 byte oder 8096 byte

Aufschlüsselung der Bytes

Byte	Bedeutung
1-20	Seitenkontrollinformation, Type=FPA_PAGE; enthält u.a. den Act-Key, der die Adresse der Seite angibt.
25	Ab diesem Byte ist für jede Seite des Realms ein Eintrag von 2 byte Länge vorhanden, der die Anzahl freier Bytes in der jeweiligen Seite angibt. Sie beträgt 0 für – Act-Key-0-Seite, Act-Key-N-Seite – FPA-Seiten, – DBTT-Ankerseiten, – DBTT-Seiten, – CALC-Seiten, da diese Seiten ganz für den jeweiligen Zweck reserviert sind und ihnen daher keine zusätzlichen Daten über die Freiplatzverwaltung zugewiesen werden können. Für leere Seiten ist der effektiv nutzbare Speicherplatz in einer Länge von 2028 byte, von 3980 byte oder von 8076 byte angegeben, da der Seitenkopf 20 Bytes in Anspruch nimmt.

Tabelle 19: Aufschlüsselung der Bytes einer FPA-Extent-Seite der Länge 2048 byte oder einer FPA-Seite der Länge 4000 byte oder 8096 byte

7.4 DBTT-Seiten

UDS/SQL benötigt pro Satzart eine Tabelle - genannt DBTT (Database Key Translation Table) - in der alle aktuell gespeicherten Sätze über ihren Database Key verwaltet werden. Dieser ist im Prinzip ein Index, welcher innerhalb der DBTT auf die dort stehende physische Adresse des Satzes in der Datenbank verweist. Im Folgenden wird der Aufbau der DBTT-Ankerseiten und der eigentlichen DBTT-Seiten beschrieben.

7.4.1 DBTT-Ankerseite

Nur für den SSIA-RECORD gibt es keine DBTT-Ankerseite. Bei allen anderen Satzarten wird die DBTT über DBTT-Ankerseiten verwaltet. Die erste DBTT-Ankerseite ist im BPSIA-Protokoll vermerkt. Um die größtmögliche DBTT einer Satzart zu verwalten - in einer 4KB-Datenbank bedeutet das über 2 Milliarden DB-Keys - benötigt man ggf. mehrere miteinander verkettete DBTT-Ankerseiten. DBTT-Ankerseiten liegen in demjenigen Realm, in dem die DBTT liegt. Sie werden nur in der aktuell benötigen Länge angelegt. In den allermeisten Fällen wird pro Satzart nicht mehr als eine DBTT-Ankerseite benötigt. Die DBTT selbst besteht aus der DBTT-Basis mit einer variablen Anzahl von DBTT-Seiten und ggf. DBTT-Extents, bestehend jeweils aus 128 hintereinanderliegenden Pam-Seiten.

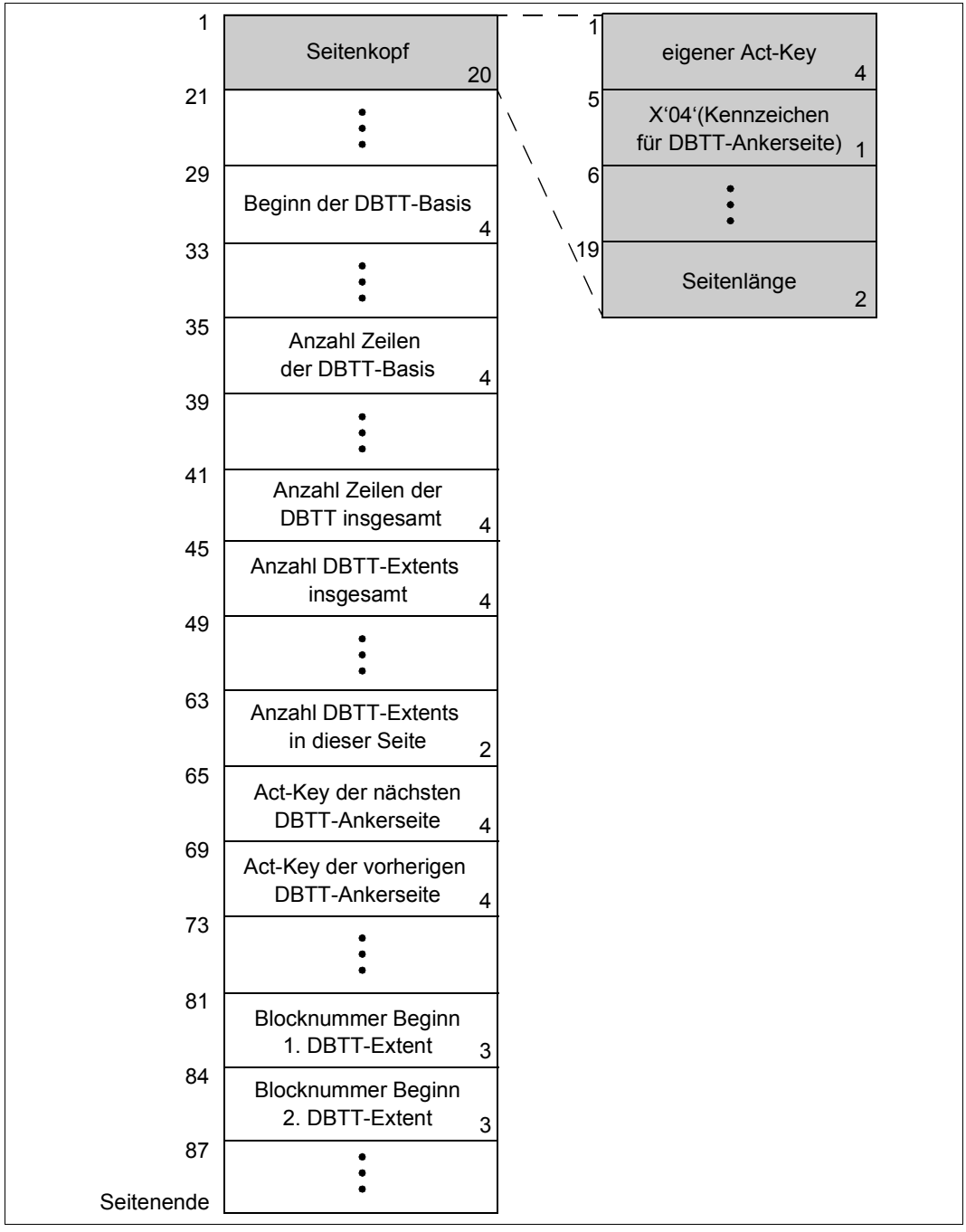


Bild 50: Aufbau einer DBTT-Ankerseite

7.4.2 DBTT-Seite

Dieser Abschnitt beschreibt das Layout einer DBTT-Seite, sowie den Aufbau der DBTT-Einträge nach Zeilen und Spalten. Zu jedem Satz einer Satzart existiert genau eine DBTT-Zeile.

Wenn die Satzart keine Ownersatzart ist, besteht die DBTT nur aus der Spalte 0, die die Act-Keys der Sätze enthält.
Im anderen Fall kommt eine Spalte hinzu für jeden Verweis auf

– eine Adressliste (MODE IS POINTER-ARRAY),

– eine Liste (MODE IS LIST),

– eine Sort-Key-Tabelle (MODE IS CHAIN, ORDER IS SORTED INDEXED),

– eine indizierte Set-SEARCH-Key-Tabelle.

Aus der Zeilenlänge der DBTT und dem Database-Key-Wert, der die Zeile innerhalb der DBTT angibt, und der Blocknummer von zugehörigem DBTT-Teil, DBTT-Basis oder DBTT-Extent kann UDS/SQL die Lage der zugehörigen DBTT-Zeile berechnen.

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, beträgt die Länge einer DBTT-Seite 2048 byte, 4000 byte oder 8096 byte.

Aufbau einer DBTT-Seite der Länge 2048 byte

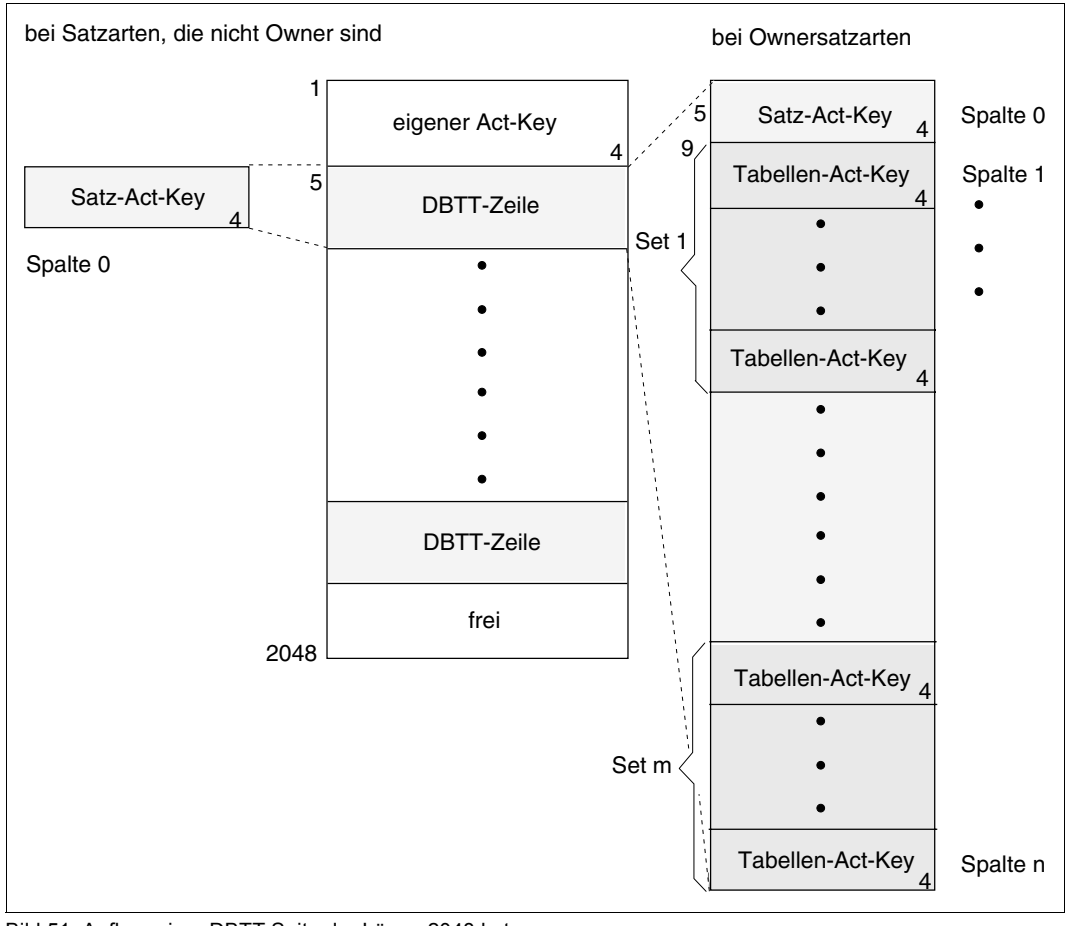


Bild 51: Aufbau einer DBTT-Seite der Länge 2048 byte

Aufbau einer DBTT-Seite der Länge 4000 byte oder 8096 byte

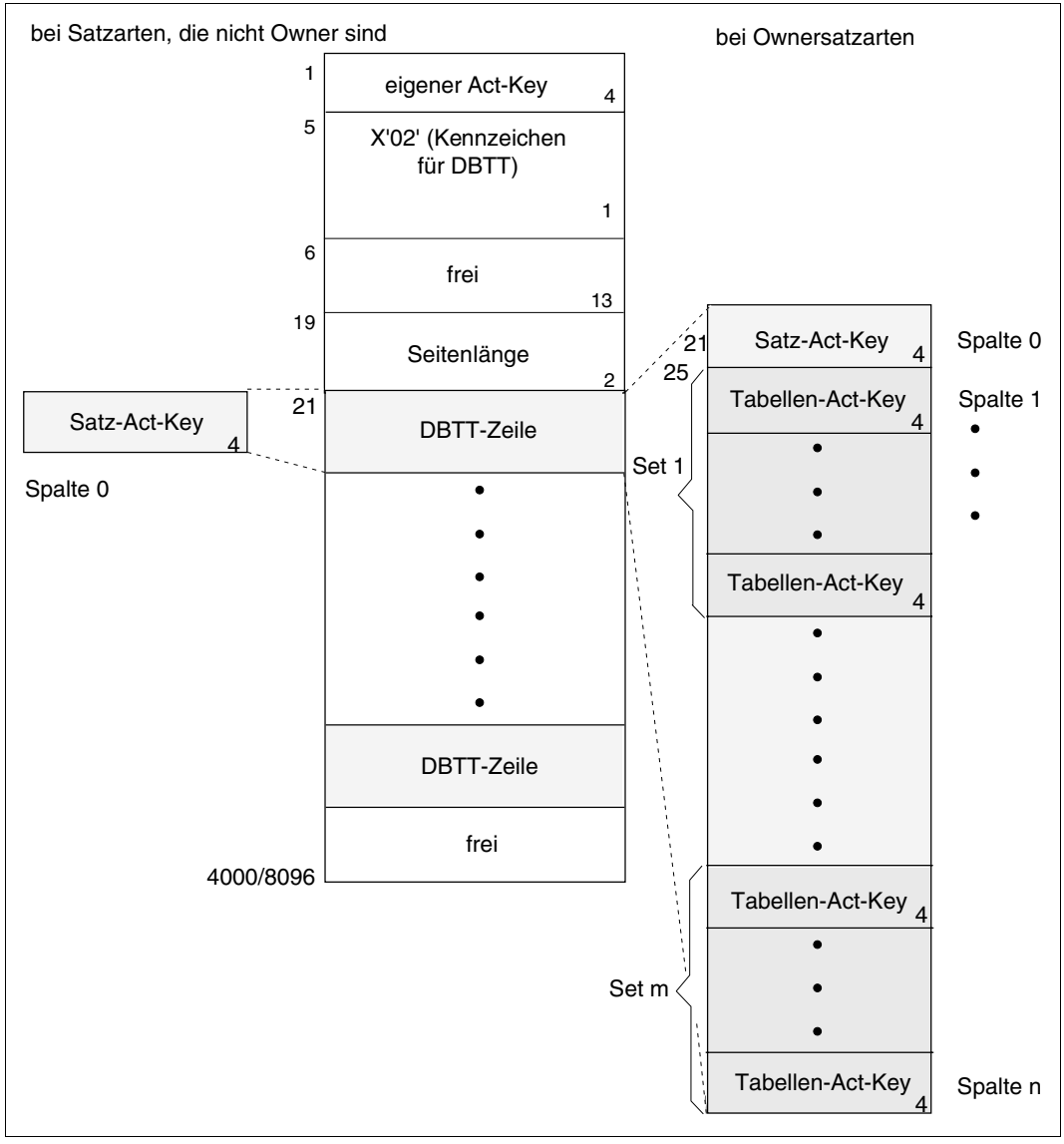


Bild 52: Aufbau einer DBTT-Seite der Länge 4000 byte oder 8096 byte

7.5 Direkte CALC-Seite

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, beträgt die Länge einer direkten CALC-Seite 2048 byte, 4000 byte oder 8096 byte.

Direkte CALC-Seiten der Länge 2048 byte unterscheiden sich von direkten CALC-Seiten der Länge 4000 byte bzw. 8096 byte hinsichtlich der Länge der Einträge für den Database-Key-Wert, die Satzfolgenummer (RSQ) und den Seitenindex:

– Bei direkten CALC-Seiten der Länge 2048 byte beträgt die Länge des Eintrags für den Database-Key-Wert 4 byte, die Länge des RSQ-Eintrags 3 byte und die Länge des Seitenindex-Eintrags 8 byte.

– Bei direkten CALC-Seiten der Länge 4000 byte oder 8096 byte beträgt die Länge des Eintrags für den Database-Key-Wert 8 byte, die Länge des RSQ-Eintrags 6 byte und die Länge des Seitenindex-Eintrags 12 byte.

In Bild 53 sind die Angaben „zahl1/zahl2“ (z.B. 25/29) wie folgt zu lesen:

– „zahl1“ gibt den Wert an, der für direkte CALC-Seiten der Länge 2048 byte gilt.

– „zahl2“ gibt den Wert an, der für direkte CALC-Seiten der Länge 4000 byte bzw. 8096 byte gilt.

Alle anderen Längen- und Distanzangaben gelten gleichermaßen für direkte CALC-Seiten der Länge 2048 byte und direkte CALC-Seiten der Länge 4000 byte bzw. 8096 byte.

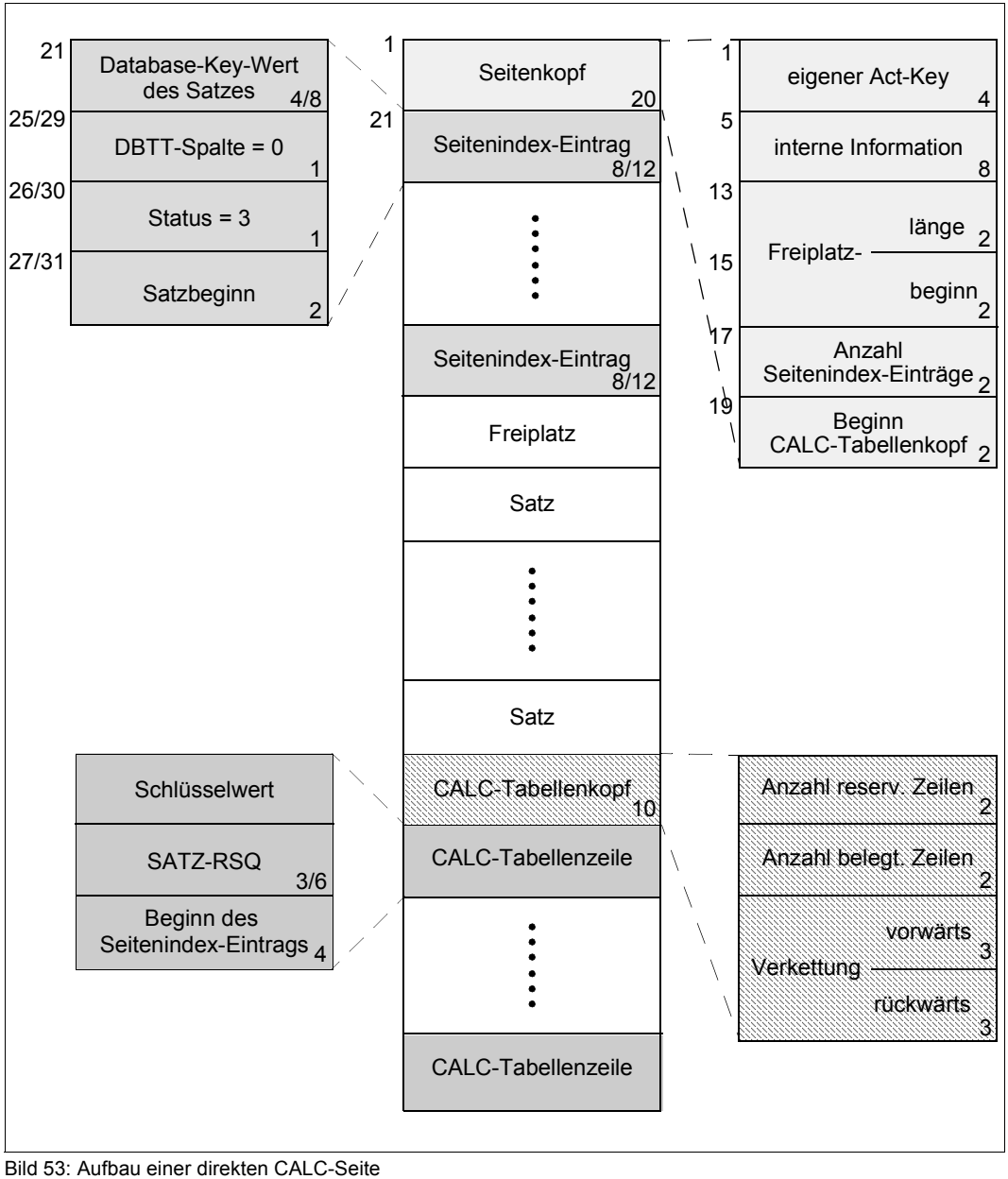


Bild 53: Aufbau einer direkten CALC-Seite

Aufschlüsselung der Bytes 1-28 bzw. 1-32

Byte	Bedeutung
1-4	gibt die Adresse der Seite an.
13-16	gibt die Länge und den Beginn des Freiplatzes an. Da der Freiplatz beim CALC-Tabellenkopf beginnend mit Sätzen aufgefüllt wird, grenzt der Satz, der zuerst in die Seite gespeichert wird, an den CALC-Tabellenkopf. Der Freiplatz beginnt hinter dem letzten Seitenindex-Eintrag und endet vor dem ersten Satz. Diese Freiplatzangaben bilden die zweite Stufe der insgesamt dreistufigen Freiplatzverwaltung.
17-18	gibt die Anzahl der Seitenindex-Einträge und damit die Anzahl der in der Seite enthaltenen Sätze an.
19-20	zeigt auf den Beginn des CALC-Tabellenkopfes.
21-24 bzw. 21-28	gibt den Database-Key-Wert des zuerst in die Seite gespeicherten Satzes an.
25 bzw. 29	Da sich die Seitenindex-Einträge in CALC-Seiten immer auf Sätze beziehen, ist die zugehörige DBTT-Spalte die Spalte 0.
26 bzw. 30	Status=3 identifiziert den zum Seitenindex-Eintrag gehörigen Satz als CALC-Satz.
27-28 bzw. 31-32	zeigt auf den zuerst eingespeicherten CALC-Satz.

Tabelle 20: Aufschlüsselung der Bytes 1-28 bzw. 1-32 der CALC-Seite

Über die Felder „Verkettung“ werden überlaufende Seiten mit ihren Überlaufseiten verkettet.

UDS/SQL berechnet aus der Satzlänge und der Länge des Schlüsselfeldes, wie viele Sätze maximal in der Seite untergebracht werden können. Daraufhin wird am Ende der Seite eine CALC-Tabelle mit der berechneten Anzahl von Tabellenzeilen eingerichtet. Jede Zeile der CALC-Tabelle verweist auf genau einen Satz. Die Tabelleneinträge sind nach den Schlüsselwerten aufsteigend sortiert, bei gleichen Schlüsselwerten nach RSQs.

Außerdem gibt es zu jedem Satz genau einen Seitenindex-Eintrag.
Wenn UDS/SQL den Schlüsselwert eines Satzes kennt, findet es den Satz in der Seite über die CALC-Tabelle und den zugehörigen Seitenindex-Eintrag. Kennt UDS/SQL den Database-Key-Wert, so findet es den Satz allein über den Seitenindex-Eintrag.

7.6 Indirekte CALC-Seite

Indirekte CALC-Seiten entstehen für eine Satzart, die mit LOCATION MODE IS CALC gespeichert wird, wenn die Satzart gleichzeitig mit der SSL auf eine der folgenden Arten definiert ist:

– als Member mit MODE IS LIST
– mit PLACEMENT OPTIMIZATION
– mit COMPRESSION FOR ALL ITEMS

Bei SEARCH KEY USING CALC entstehen immer indirekte CALC-Seiten.

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, beträgt die Länge einer indirekten CALC-Seite 2048 byte, 4000 byte oder 8096 byte.

Indirekte CALC-Seiten der Länge 2048 byte unterscheiden sich von indirekten CALC-Seiten der Länge 4000 byte bzw. 8096 byte hinsichtlich der Länge der Einträge für die Satzfolgenummer (RSQ):

– Bei indirekten CALC-Seiten der Länge 2048 byte ist der RSQ-Eintrag 3 byte lang.

– Bei indirekten CALC-Seiten der Länge 4000 byte oder 8096 byte ist der RSQ-Eintrag 6 byte lang.

Von direkten CALC-Seiten unterscheiden sich indirekte CALC-Seiten im Aufbau darin, dass sie keine Sätze und keine Seitenindexeinträge enthalten. Die CALC-Tabellenzeile enthält deshalb den Probable Position Pointer (PPP) auf den zugehörigen Satz. Ansonsten gilt die Beschreibung der direkten CALC-Seiten auch für die indirekten CALC-Seiten.

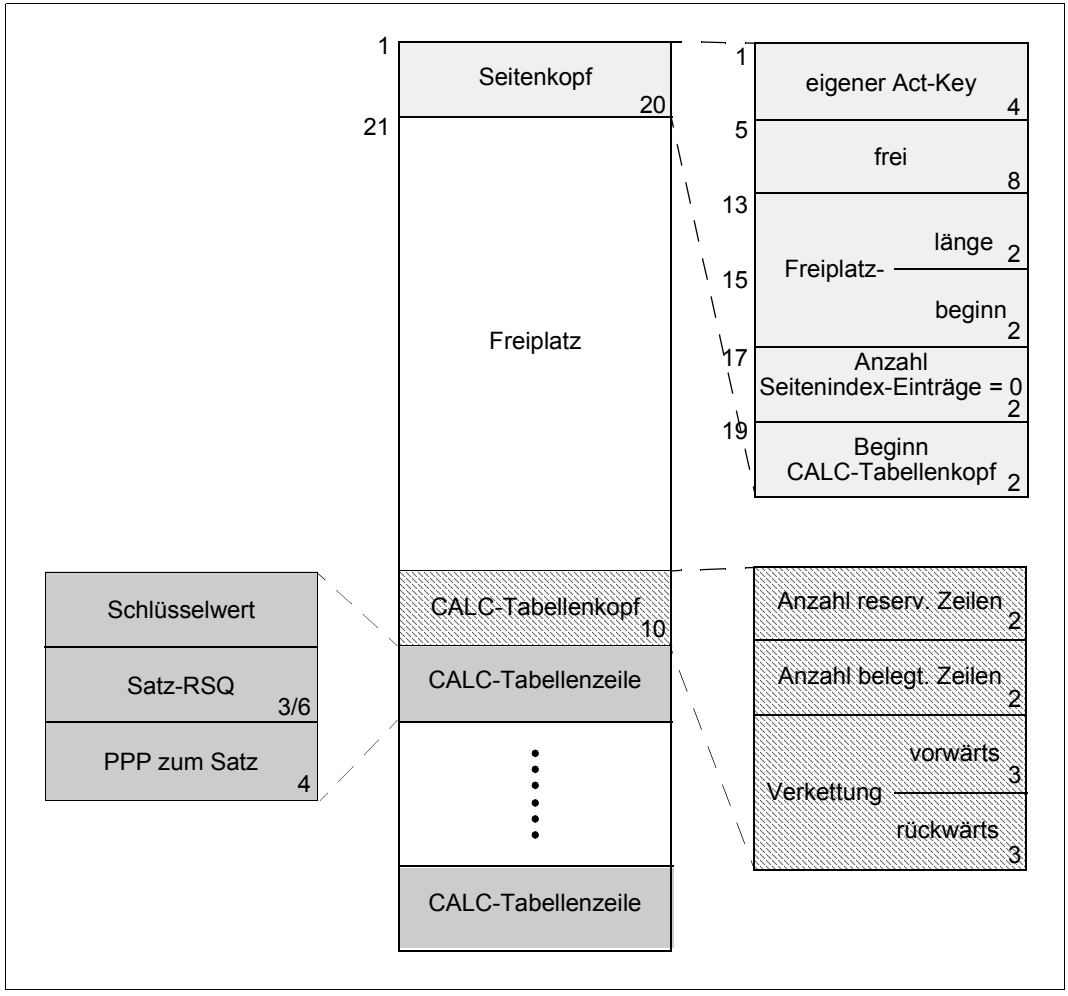


Bild 54: Aufbau einer indirekten CALC-Seite

7.7 Datenseite

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, beträgt die Länge einer Datenseite 2048 byte, 4000 byte oder 8096 byte.

Datenseiten der Länge 2048 byte unterscheiden sich von Datenseiten der Länge 4000 byte oder 8096 byte hinsichtlich der Länge der Einträge für den Database-Key-Wert und den Seitenindex:

– Bei Datenseiten der Länge 2048 byte ist der Eintrag für den Database-Key-Wert 4 byte lang und der Seitenindex-Eintrag 8 byte lang.

– Bei Datenseiten der Länge 4000 byte oder 8096 byte ist der Eintrag für den Database-Key-Wert 8 byte lang und der Seitenindex-Eintrag 12 byte lang.

Folgende Daten werden in eine Datenseite aufgenommen:

– Adresslisten
– Listen
– Sort-Key-Tabellen
– SEARCH-Key-Tabellen
– Sätze, die nicht in einen Hashbereich gespeichert werden

In Bild 55 sind die Angaben „zahl1/zahl2“ (z.B. 25/29) wir folgt zu lesen:

– „zahl1“ gibt den Wert an, der für Datenseiten der Länge 2048 byte gilt.
– „zahl 2“ gibt den Wert an, der für Datenseiten der Länge 4000 byte oder 8096 byte gilt.

Alle anderen Längen- und Distanzangaben gelten gleichermaßen für Datenseiten der Länge 2048 byte und Datenseiten der Länge 4000 byte bzw. 8096 byte.

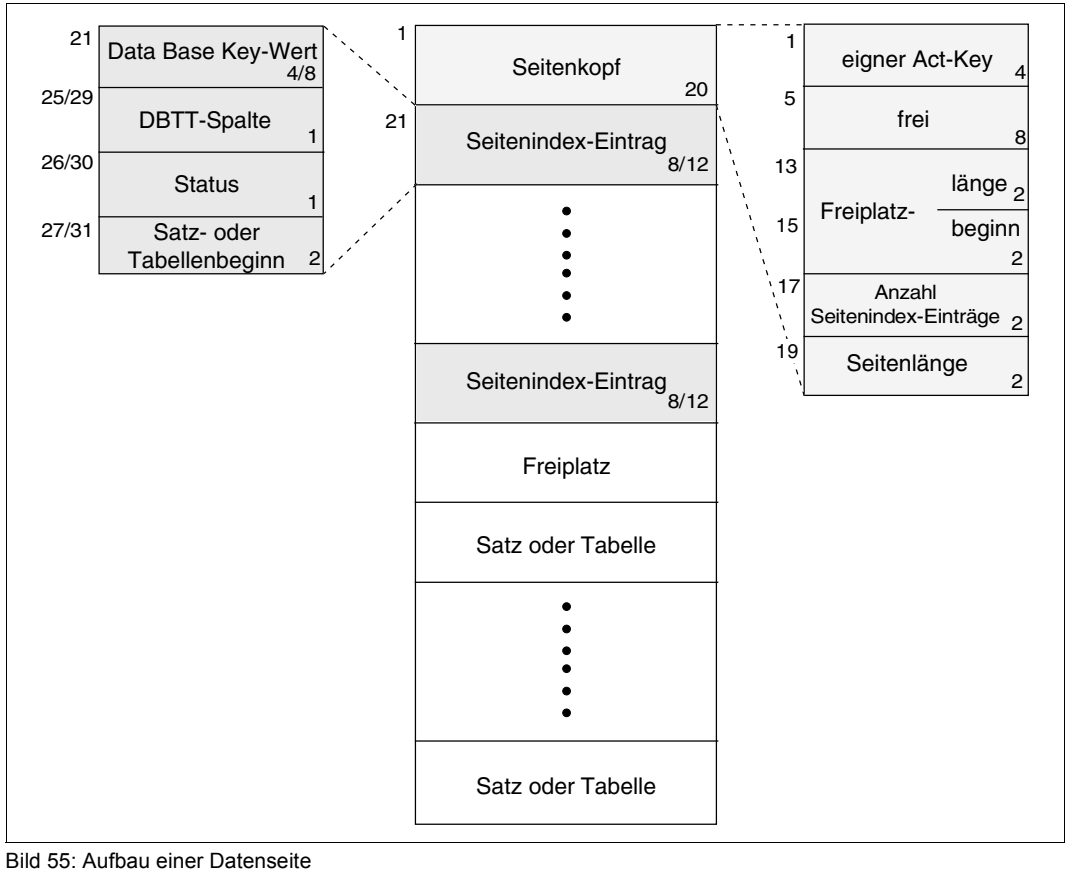


Bild 55: Aufbau einer Datenseite

Aufschlüsselung der Bytes 1-28 bzw. 1-32

Byte	**Bedeutung**	
1-4	gibt die Adresse der Seite an.	
13-16	gibt die Länge und den Beginn des Freiplatzes an. Da der Freiplatz am Seitenende beginnend mit Sätzen bzw. Tabellen aufgefüllt wird, grenzt der Beginn des Freiplatzes an den zuletzt eingespeicherten Satz bzw. die zuletzt eingespeicherte Tabelle. Diese Freiplatzangaben bilden die zweite Stufe der insgesamt dreistufigen Freiplatzverwaltung.	
17-18	gibt die Anzahl der Seitenindex-Einträge an.	
19-20	Die Seitenlänge beträgt 2048 byte, 4000 byte oder 8096 byte.	
	Satz	Tabelle
21-24 bzw. 21-28	Database-Key-Wert des Satzes	Database-Key-Wert des zugehörigen Ownersatzes
25/29	DBTT-Spalte=0	DBTT-Spalte=1-n
26/30	Status=0: Satz, der nicht zu einer Liste gehört (auch Ankersatz) Status=2: Satz, der zu einer Liste gehört	Status=1
27-28 bzw. 31-32	zeigt auf den Beginn des Satzes bzw. der Tabelle.	

Tabelle 21: Aufschlüsselung der Bytes 1-28 bzw. 1-32 einer Datenseite

Zu jedem Satz und zu jeder Tabelle in der Seite gibt es einen Seitenindex-Eintrag, der die Position der Tabelle bzw. des Satzes angibt.
Bei Listen gibt es sowohl einen Seitenindex-Eintrag für die Liste, als auch für jeden darin enthaltenen Satz. Insbesondere ist die logische Reihenfolge von Sätzen in einer Liste nicht durch die physische Reihenfolge der Sätze, sondern der Seitenindex-Einträge gegeben.

Eine Datenseite kann nicht gleichzeitig zwei Ankersätze enthalten.

8 Aufbau der Sätze und Tabellen

Um eine Datenbank mit DDL und SSL aufbauen zu können, benötigen Sie im Allgemeinen keine Informationen über den Aufbau von Sätzen und Tabellen.
Dieses Kapitel dient zum Nachschlagen, wenn Sie an speziellen Details interessiert sind.

8.1 Aufbau der Sätze

Benutzerdefinierter Satz

Der physische Aufbau eines Satzes, den Sie mit der Schema-DDL definiert haben, beinhaltet meist mehr als nur die Felder, die Sie für die Satzart vereinbart haben: In den meisten Fällen wird den Sätzen beim Speichern automatisch eine SCD (Set-Connection-Data) angegliedert. Die SCD kann folgende Bestandteile enthalten:

– die Zeiger, die UDS/SQL zur Speicherung einer Set-Occurrence als Kette benötigt
– den Zeiger vom Owner auf eine zugehörige Adressliste oder Liste
– den Zeiger vom Member zum zugehörigen Owner
– die Anzeige, ob der Satz in einem SYSTEM-Set eingehängt ist

Wenn Sie nicht für eine Satzart ein variables Feld oder die komprimierte Speicherung definiert haben, werden die Sätze der Satzart in der folgenden Form gespeichert:

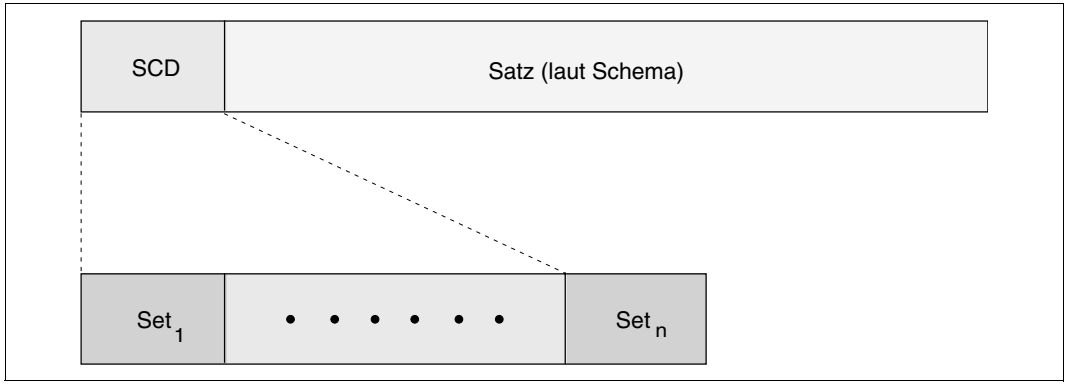


Bild 56: Normalform eines benutzerdefinierten Satzes

Andernfalls gilt Folgendes:

– Unkomprimierte Speicherung im Falle STORE Format-1 und Subschema = Schema.
  Das bedeutet: Es gibt den Kompressionskopf, aber keine Kompressionseinträge, die Anzahl der Einträge im Kopf ist 0, der Satz wird wie bei der unkomprimierten Form vollständig gespeichert.

– Komprimierte Speicherung in allen anderen Fällen.
  Das bedeutet: In Format-2 weggelassene und/oder nicht im Subschema vorhandene Felder werden im Gegensatz zur unkomprimierten Form nicht mit binär 0 belegt, sondern ganz weggelassen, die übrig bleibenden Teilstrings werden in den Kompressionseinträgen verwaltet. Die Kompressionseinträge sind nach absteigenden Distanzen sortiert, die Länge der Teilstrings ergibt sich aus der Differenz des Inhalts ("vor Komp") zweier aufeinanderfolgender Kompressionseinträge, beim ersten Kompressionseintrag ist der Bezugswert bzgl. Länge "Satzlänge" im Kompressionskopf.

– Variables Feld:
  Komprimierte Speicherung mit genau einem Kompressionseintrag. Das variable Feld selbst wird nicht im gespeicherten Satz verwaltet, sondern über die Item String List des SSIA-Protokolls, die Satzlänge im Kompressionskopf ist die aktuelle Länge inklusive SCD, Kompressionskopf und des einen Kompressionseintrags, dieser verwaltet den gesamten Satz inklusive des variablen Teils.

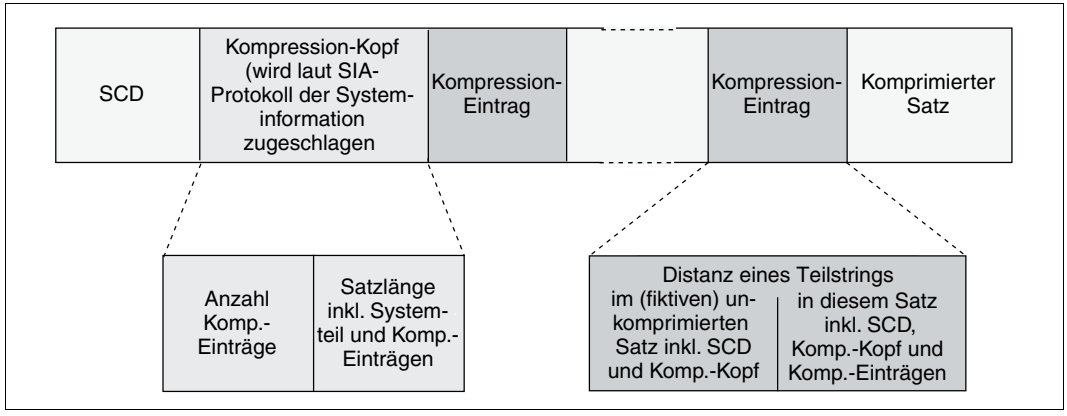


Bild 57: Komprimierte Form eines benutzerdefinierten Satzes

Ankersatz

Außer den Sätzen, die Sie in der Schema-DDL definieren, gibt es für jeden SYSTEM-Set einen Ankersatz, den UDS/SQL automatisch erzeugt. Ein Ankersatz übernimmt bei SYSTEM-Sets die Funktion, die die DBTT einer Ownersatzart bei normalen Sets hat. Ein Ankersatz enthält seinen eigenen Act-Key und die Act-Keys der zum SYSTEM-Set zugehörigen Tabellen. Er wird um SCD erweitert, wenn die Sätze des SYSTEM-Set als Kette gespeichert werden.

Set-Connection-Data

UDS/SQL fügt einem Satz beim Einspeichern automatisch eine SCD (Set-Connection-Data) an, wenn er mit anderen Sätzen oder Tabellen verknüpft werden muss. Der Tabelle 22 entnehmen Sie, wie diese Verknüpfungsinformation im Einzelnen aufgebaut ist.

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, kann die SCD unterschiedlich lang sein. Wo sich die Seitenlänge der Datenbank auf die Länge der SCD auswirkt, enthält in Tabelle 22 die Spalte „SCD-Länge in byte“ zwei Werte (z.B. 8/12):

– Der erste Wert gibt die Länge der SCD in einer Datenbank mit 2048 byte Seitenlänge an.
– Der zweite Wert gibt die Länge der SCD in einer Datenbank mit 4000 byte bzw. 8096 byte Seitenlänge an.

MODE IS	SATZART IST	OWNER IS SYSTEM	ORDER IS LAST	LINKED TO PRIOR	MEMBER LINKED	PHYSI-CAL LINK	SCD Länge in byte	SCD bein-haltet
CHAIN	Owner	J/N	N J J/N	N N J	-	-	8/12 16/24 16/24	1 1,2 1,2
	Member	J		N J	-	-	8/12 16/24	3 3,4
		N		N N J J	N J N J	-	11/18 15/22 19/30 23/34	3,5 3,5,6 3,4,5 3,4,5,6
POINTER-ARRAY oder LIST	Owner	J		-	-	-	0	-
		N		-	-	N J	0 4	 7
	Member	J		-	-	-	1	8
		N		-	N J	-	3/6 7/10	5 5,6

Tabelle 22: Übersicht über die möglichen SCD-Kombinationen pro Set

```
1  Database-Key-Wert + PPP des ersten Membersatzes            ... 8/12 byte

2  Database-Key-Wert + PPP des letzten Membersatzes           ... 8/12 byte

3  Database-Key-Wert + PPP des folgenden Satzes               ... 8/12 byte

4  Database-Key-Wert + PPP des vorhergehenden Satzes          ... 8/12 byte

5  RSQ des Owners                                              ... 3/6 byte

6  PPP des Owners                                                ... 4 byte

7  PPP des Tabellenanfangs                                       ... 4 byte

8  Kennzeichen für SYSTEM-Set                                    ... 1 byte
   Dieses Byte enthält X'00', wenn der Membersatz zum Set gehört,
   und X'FF', wenn er nicht zum Set gehört.
   Im Fall „MODE IS CHAIN“ ist das Byte auf 3) (siehe oben) redefiniert.
```

8.2 Aufbau der Tabellen

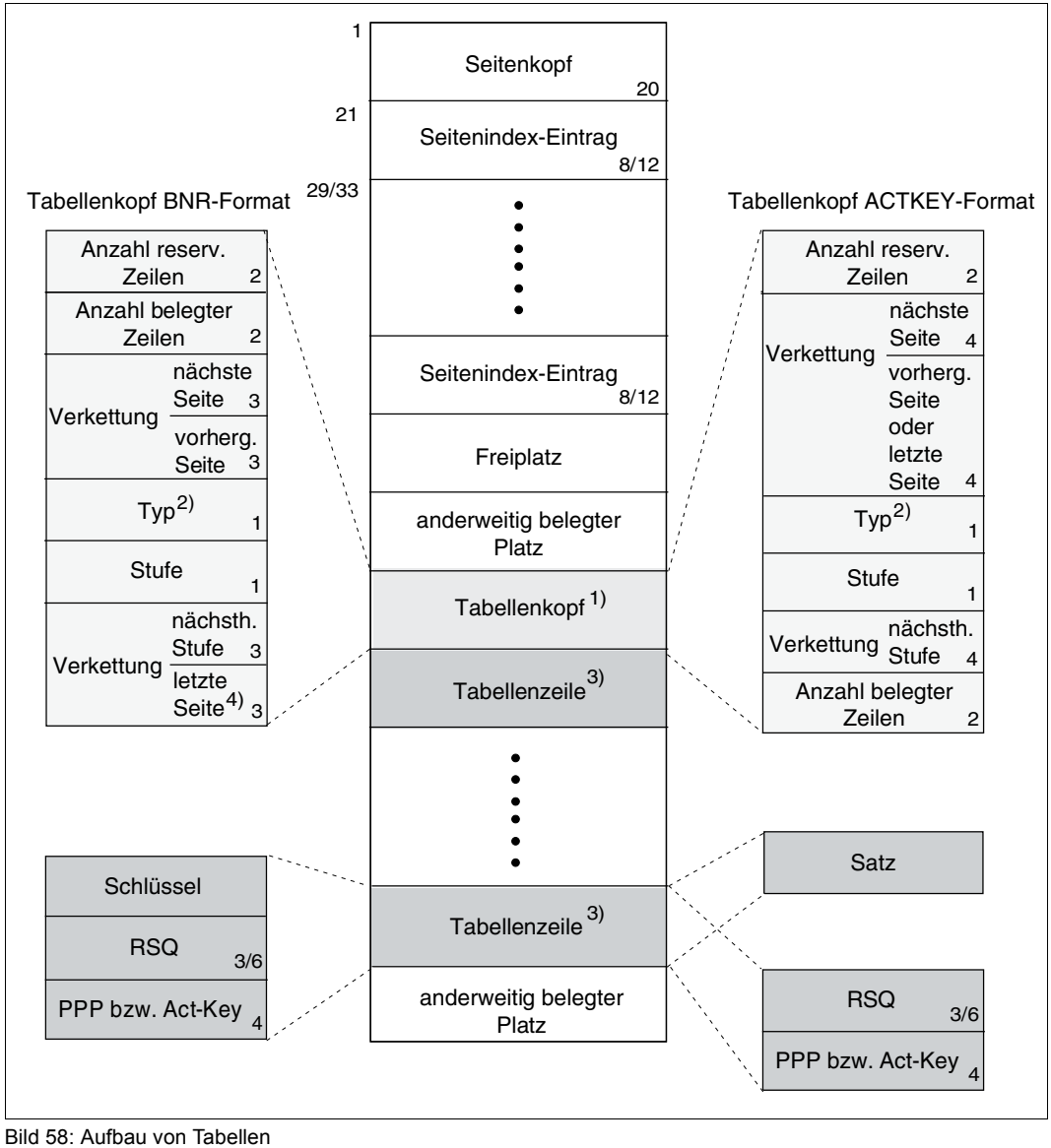


Bild 58: Aufbau von Tabellen

Erklärung zum Bild 58:

1) Der Tabellenkopf befindet sich bei Listen nicht vor den einzelnen Tabellenzeilen; im Allgemeinen liegt der Tabellenkopf hinter den Tabellenzeilen, also den Sätzen der Liste; die Zugehörigkeit der Sätze zur Liste ergibt sich aus den Seitenindexeinträgen, die hintereinander liegen und mit dem Seitenindexeintrag für den Tabellenkopf beginnen.
   Es existieren zwei unterschiedliche Formate für den Tabellenkopf:
   - BNR-Format; die Verweise im Tabellenkopf sind durch die Blocknummern (BNR) eindeutig.
   - ACTKEY-Format für über mehrere Realms verteilbare Tabellen; die Verweise enthalten neben der Blocknummer auch die Area-Reference der verknüpften Tabellenseite.

2) Bit 2^7=1: Liste
   Bit 2^6=1: mehrstufige Tabelle
   Bit 2^5=1: Tabelle ATTACHED TO OWNER
   Bit 2^4=1: Duplikat-Tabelle
   Bit 2^3=1: Tabelle im ACTKEY-Format
   Bit 2^2=1: Tabelle im ACTKEY-Format mit Verkettung zur letzten Seite.
   Dieses Bit zeigt an, daß der ACTKEY im überlagerten Feld auf die logisch letzte Stufe-0-Seite zeigt. Es darf nur in Verbindung mit Bit 2^3 gesetzt sein.

3) gilt nicht für Duplikat-Tabellen (siehe Bild 60 und Bild 61).

4) gilt nur in der höchsten Tabellenseite einer mehrstufigen Tabelle oder in der ersten Tabellenseite einer einstufigen Tabelle.

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, sind die Einträge für die Satzfolgenummer (RSQ) und den Seitenindex jeweils unterschiedlich lang:

- In Tabellen einer Datenbank mit 2048 byte Seitenlänge ist der RSQ-Eintrag 3 byte lang und der Seitenindex-Eintrag 8 byte lang.
- In Tabellen einer Datenbank mit 4000 byte oder 8096 byte Seitenlänge ist der RSQ-Eintrag 6 byte lang und der Seitenindex-Eintrag 12 byte lang.

Adresslisten, Listen, Sort-Key-Tabellen und standardmäßige SEARCH-Key-Tabellen sind nach dem im Bild 58 dargestellten Prinzip aufgebaut. (Für Duplikat-Tabellen gilt die Darstellung der Tabellenzeile nicht). Sie werden immer in Datenseiten gespeichert. Wenn eine Tabelle nicht eine ganze Seite ausfüllt, kann der übrige Platz auch mit Sätzen oder weiteren Tabellen belegt sein. Im Tabellenkopf werden die Platzreservierungen für die Tabelle durch-

geführt, die Sie in den POPULATION-Klauseln definieren. Dies ist die dritte Stufe der insgesamt dreistufigen Freiplatzverwaltung. Außerdem enthält der Tabellenkopf die Verkettung zu anderen Tabellenteilen, wenn die Tabelle größer als eine Seite ist, und die Verkettung zur nächsthöheren Tabellenstufe, wenn die Seite zu einer untergeordneten Stufe gehört. Zwei weitere Kennzeichen sind vorhanden für die Tabellenstufe, der die Seite selbst angehört und für den Tabellentyp. Das ACTKEY-Format des Tabellenkopfes wird in verteilbaren Listen genutzt.

Welche der drei möglichen Tabellenzeilen zu welchem Tabellentyp gehört, entnehmen Sie der folgenden Übersicht:

DDL/SSL-Klauseln	**Tabellentyp**	**Inhalt der Tabellenzeile**		**Name der Tabelle**
		unterste Stufe	**übergeordnete Stufe**	
MODE IS POINTER-ARRAY ORDER IS LAST/ FIRST/NEXT/PRIOR	einstufig	RSQ, PPP	-	einstufige Adressliste
MODE IS POINTER-ARRAY ORDER IS SORTED INDEXED BY DATABASE-KEY	mehrstufig	RSQ, PPP	RSQ, Act-Key	mehrstufige Adressliste
MODE IS POINTER-ARRAY ORDER IS SORTED INDEXED BY DEFINED KEYS ASCENDING/ DESCENDING KEY IS...	mehrstufig	Schlüsselwert, RSQ, PPP	Schlüsselwert, RSQ, Act-Key	
MODE IS LIST ORDER IS LAST/ FIRST/NEXT/PRIOR	einstufig	Membersatz	-	einstufige Liste
MODE IS LIST ORDER IS SORTED INDEXED BY DATABASE-KEY	mehrstufig	Membersatz	RSQ, Act-Key	mehrstufige Liste
MODE IS LIST ORDER IS SORTED INDEXED BY DEFINED KEYS ASCENDING/ DESCENDING KEY IS...	mehrstufig	Membersatz	Schlüsselwert, RSQ, Act-Key	

Tabelle 23: Übersicht über die Tabellentypen (Teil 1 von 2)

DDL/SSL-Klauseln	Tabellentyp	Inhalt der Tabellenzeile		Name der Tabelle
		unterste Stufe	übergeordnete Stufe	
MODE IS CHAIN ORDER IS SORTED INDEXED BY DATABASE-KEY	mehrstufig	RSQ, PPP	RSQ, Act-Key	Sort-Key-Tabelle
MODE IS CHAIN ORDER IS SORTED INDEXED BY DEFINED KEYS ASCENDING/ DESCENDING KEY IS...	mehrstufig	Schlüsselwert, RSQ, PPP	Schlüsselwert, RSQ, Act-Key	
SEARCH KEY IS... USING INDEX TYPE IS REPEATED-KEY (auf Set- oder Satzartebene)	mehrstufig	Schlüsselwert, RSQ, PPP	Schlüsselwert, RSQ, Act-Key	SEARCH-Key-Tabelle
SEARCH KEY IS... USING INDEX TYPE IS DATABASE-KEY-LIST	mehrstufig	siehe Bild 60 und Bild 61	Schlüsselwert, ACT_KEY	Duplikattabelle

Tabelle 23: Übersicht über die Tabellentypen (Teil 2 von 2)

Die Tabellenzeilen einer übergeordneten Tabellenstufe verweisen mit Act-Keys auf die zugehörigen Seiten der nächstniedrigeren Tabellenstufe.
Die Tabellenzeilen der untersten Tabellenstufe verweisen mit Probable Position Pointer (PPP) auf die Seiten, die die zugehörigen Sätze enthalten. Wenn ein Probable Position Pointer nicht aktuell ist, sucht UDS/SQL den Satz mit Hilfe der RSQ über die DBTT.

Das folgende Bild 59 stellt den hierarchischen Aufbau einer mehrstufigen Tabelle dar. Das Beispiel trifft auf eine Adressliste, auf eine Sort-Key-Tabelle und auf eine SEARCH-Key-Tabelle zu.
Bei einer Liste enthält die Stufe 0 die Sätze selbst.

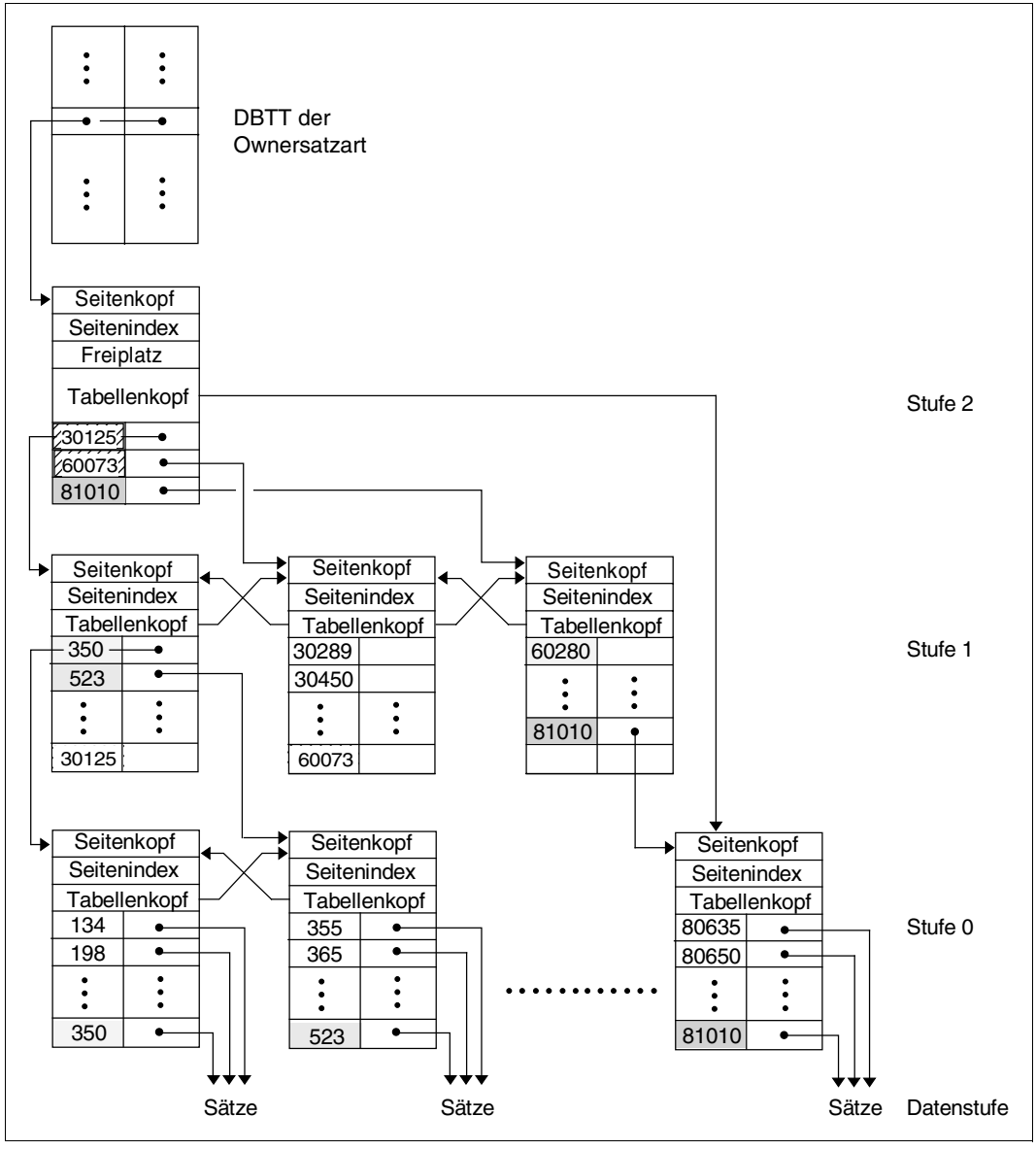


Bild 59: Hierarchischer Aufbau einer mehrstufigen Tabelle

Duplikat-Tabelle

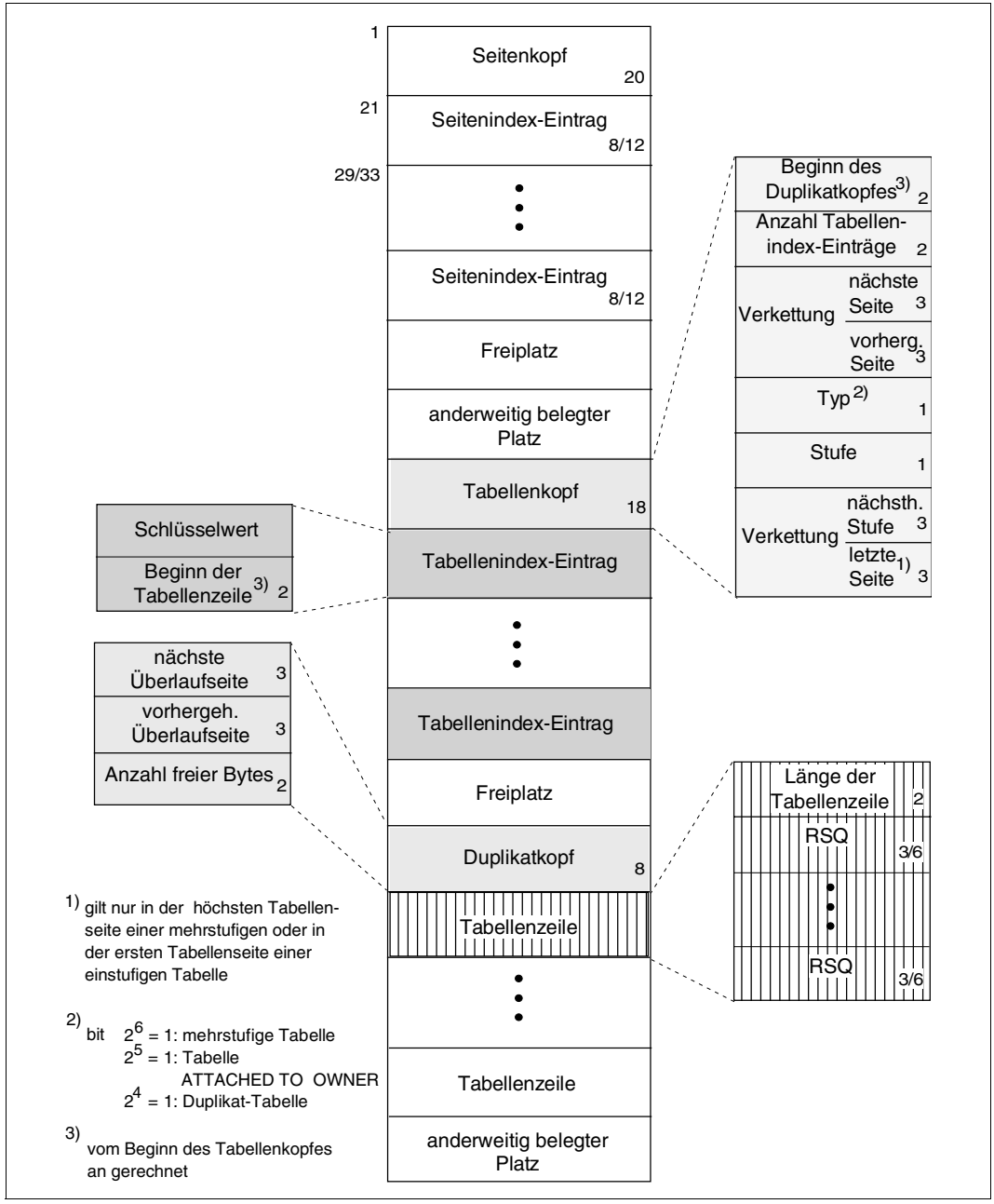


Bild 60: Aufbau einer Duplikat-Tabelle

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, sind die Einträge für Database-Key-Wert und die Satzfolgenummer (RSQ) sowie die Seitenindex-Einträge jeweils unterschiedlich lang:

– In Tabellen einer Datenbank mit 2048 byte Seitenlänge ist der Eintrag für den Database-Key-Wert 4 byte lang, der RSQ-Eintrag 3 byte lang und der Seitenindex-Eintrag 8 byte lang.

– In Tabellen einer Datenbank mit 4000 byte oder 8096 byte Seitenlänge ist der Eintrag für den Database-Key-Wert 8 byte lang, der RSQ-Eintrag 6 byte lang und der Seitenindex-Eintrag 12 byte lang.

Überlaufseite einer Duplikat-Tabelle

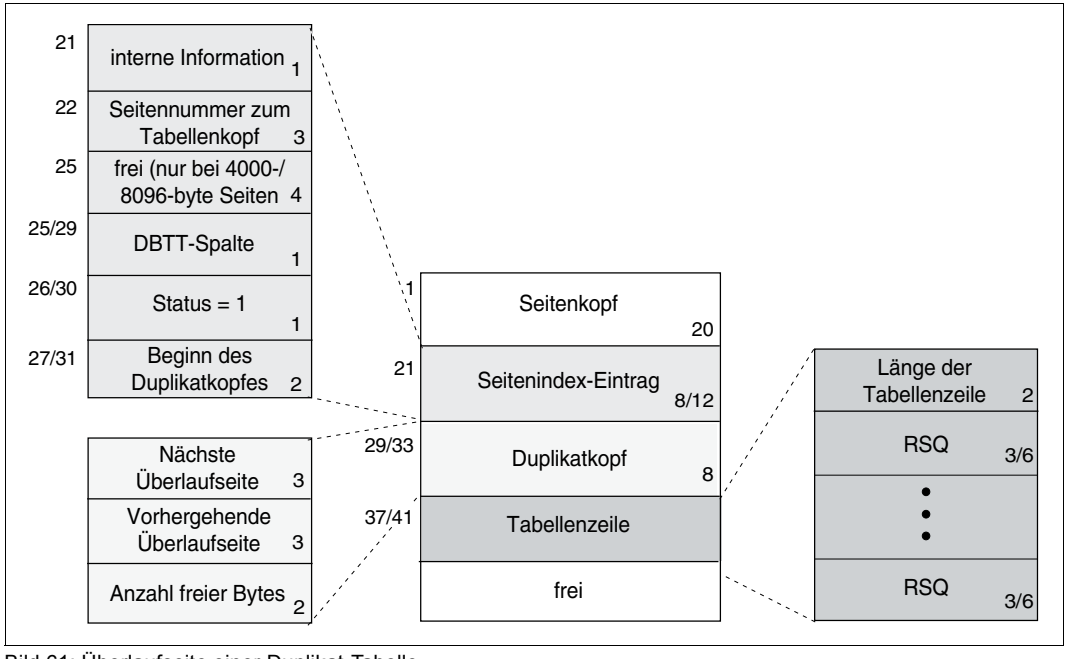


Bild 61: Überlaufseite einer Duplikat-Tabelle

Je nachdem, welche Seitenlänge für die Datenbank festgelegt ist, beträgt die Länge einer Überlaufseite 2048 byte, 4000 byte oder 8096 byte.
Für die Länge der Einträge zu Satzfolgenummer (RSQ) und Seitenindex und gilt:

– In einer 2048 byte-Überlaufseite ist der RSQ-Eintrag 3 byte lang und der Seitenindex-Eintrag 8 byte lang.

– In einer 4000 byte- oder 8096 byte-Überlaufseite ist der RSQ-Eintrag 6 byte lang und der Seitenindex-Eintrag 12 byte lang.

Duplikat-Tabellen sind spezielle SEARCH-Key-Tabellen, in denen ein mehrfach auftretender Schlüsselwert nur jeweils einmal gespeichert wird. Wenn ein Teil der Duplikat-Tabelle, der nur einen Schlüsselwert enthält, über die Größe einer Seite erweitert werden muss, dann legt UDS/SQL eine Überlaufseite an. In dieser Überlaufseite wird die zum Schlüssel gehörige Tabellenzeile fortgesetzt. Die Verbindung zur Überlaufseite wird nicht über den Tabellenkopf, sondern über den Duplikatkopf hergestellt.

9 Nachschlageteil

In den vorangehenden Kapiteln konnten Sie sich mit der Bedeutung und den Verwendungsmöglichkeiten der Schema-DDL-, SSL- und Subschema-DDL-Klauseln vertraut machen.
In diesem Kapitel finden Sie die Syntaxregeln, die Sie beachten müssen, um die jeweilige Sprache formal fehlerfrei benutzen zu können.
Die verwendete Metasprache ist auf der Seite 18 beschrieben.

Allgemeine Syntaxregeln

variable

müssen Sie bei der Benutzung eines Formats durch einen aktuellen Wert ersetzen. Dabei sind vier Kategorien von Variablen zu unter scheiden:

variable	aktueller Wert
schemaname *subschemaname* *realmname* *setname* *satzname* *satzelementname* *datengruppenname* *vektorname* *feldname* *bezeichner* *hashroutine*	muss mit einem Buchstaben beginnen und darf aus höchstens 30 Zeichen bestehen. Die Zeichen dürfen Buchstaben, Ziffern und Bindestriche sein. Ein Bindestrich darf nicht auf einen Bindestrich folgen und nicht das letzte Zeichen sein. Der aktuelle Wert darf nicht mit einem Wahl- oder Schlüsselwort übereinstimmen. *satzelementname*, *datengruppenname*, *vektorname* und *feldname* müssen nur innerhalb einer Satzart eindeutig sein. Durch {IN | OF} *satzname* können Sie Namen außer bei der Namensvergabe qualifizieren. Alle übrigen Namen müssen in der gesamten Datenbank eindeutig sein. *subschemaname* muss innerhalb einer Datenbankkonfiguration in den ersten sechs Zeichen eindeutig sein.
literal	Das Literal muss in Anführungszeichen (im Fall QUOTE IS SINGLE in Apostrophe; siehe Handbuch „Aufbauen und Umstrukturieren“, Schema-DDL übersetzen) eingeschlossen werden. Diese zählen nicht zum Wert des Literals.
ganzzahl	setzt sich aus maximal 15 Ziffern zusammen. In TYPE-Klausel ist zusätzlich ein Minuszeichen erlaubt (siehe Seite 238, TYPE-Klausel).
maskenzeichenkette	siehe Abschnitt „Definieren eines Feldes“ auf Seite 51

Kommentar
: wird durch * in Spalte 7 gekennzeichnet. Den Text in den Spalten 8-72 fasst UDS/SQL dann als Kommentar auf.

Semikolon
: dürfen Sie wahlweise zur Trennung einzelner Klauseln angeben.

Seitenvorschub
: wird durch / in Spalte 7 gekennzeichnet.

Fortsetzungszeilen
: Eingaben über Spalte 72 hinaus können Sie in einer neuen Zeile fortsetzen, wenn diese mit einem Bindestrich in Spalte 7 beginnt.

Großbuchstaben
: Der COBOL-Compiler akzeptiert nur Großbuchstaben.

Spaltenkonventionen

Jede der drei Sprachen setzt sich aus Klauseln zusammen.
Die Klauseln müssen im allgemeinen ab Spalte 12 geschrieben werden.
Ab Spalte 8 werden geschrieben

- die jeweils erste Klausel eines Eintrags,
- die MEMBER-Klausel

und in der Subschema-DDL

- die erste Zeile einer Division,
- die erste Zeile einer Section und
- die Stufennummer 01.

Die Syntaxbeschreibung einer Sprache beginnt jeweils mit einer Übersicht über die Einträge und die darin enthaltenen Klauseln.

Reservierte Wörter

Es gelten die reservierten Wörter der verwendeten COBOL-Version, siehe Handbuch „COBOL2000 (BS2000) - Sprachbeschreibung“.

9.1 Syntax der Schema-DDL

```
                    ⎧SCHEMA NAME-Klausel
Schema-Eintrag      ⎨
                    ⎩[PRIVACY LOCK-Klausel].

                    ⎧AREA NAME-Klausel
Realm-Eintrag       ⎨
                    ⎩[TEMPORARY-Klausel].

                    ⎧RECORD NAME-Klausel
                    ⎪
                    ⎪[LOCATION MODE-Klausel]
                    ⎪
                    ⎪WITHIN-Klausel
                    ⎪
                    ⎪[SEARCH KEY-Klausel].
Satz-Eintrag        ⎨
                    ⎪Satzelementname-Klausel
                    ⎪
                    ⎪[PICTURE-Klausel]
                    ⎪
                    ⎪[TYPE-Klausel]
                    ⎪
                    ⎩[OCCURS-Klausel].

                    ⎧SET-NAME-Klausel
                    ⎪
                    ⎪[DYNAMIC-Klausel]
                    ⎪
                    ⎪ORDER-Klausel
                    ⎪
                    ⎪OWNER-Klausel.
Set-Eintrag         ⎨
                    ⎪[MEMBER-Klausel
                    ⎪
                    ⎪[ASCENDING/DESCENDING-KEY-Klausel]
                    ⎪
                    ⎪[SEARCH KEY-Klausel]
                    ⎪
                    ⎩[SET OCCURRENCE SELECTION-Klausel]].
```

Bild 62: Aufbau der Schema-DDL

Die Beschreibung der logischen Datenstruktur beginnen Sie immer mit dem Schema-Eintrag und mindestens einem Realm-Eintrag.

Für die Abfolge der Satz-, Set- und der übrigen Realm-Einträge gilt:

– Bevor Sie einen Set definieren, müssen Sie die beiden zugehörigen Satzarten definieren.

– Bevor Sie eine Satzart definieren, müssen Sie alle Realms definieren, die Sie in dessen WITHIN-Klausel nennen.

9.1.1 Der Schema-Eintrag

```
SCHEMA NAME IS schemaname
     [PRIVACY LOCK FOR COPY IS literal-1[ OR literal-2]].
```

literal-1,-2
darf maximal 10 Zeichen umfassen.

Sie benennen das Schema und vereinbaren Kennwörter, die das Schema vor dem unbefugten Erstellen eines Subschemas schützen.

9.1.2 Der Realm-Eintrag

```
AREA NAME IS realmname
    [AREA IS TEMPORARY].
```

Sie benennen einen Realm und erklären ihn ggf. zum Temporären Realm.

i

In einer Datenbank mit 2048 byte Seitenlänge können Sie maximal 123 Realms definieren.
In einer Datenbank mit 4000 byte oder 8096 byte Seitenlänge können Sie maximal 245 Realms definieren.

Sie dürfen nur *einen* Temporären Realm definieren.

9.1.3 Der Satz-Eintrag

```
RECORD NAME IS satzname
                          ⎧⎧DIRECT      ⎫⎧feldname-1 ⎧IN⎫ satzname⎫⎫
    [LOCATION MODE IS     ⎨⎩DIRECT-LONG⎭⎩            ⎩OF⎭        ⎭⎬]
                          ⎪bezeichner-1                            ⎪
                          ⎪CALC[ hashroutine] USING feldname-2,...  ⎪
                          ⎩     DUPLICATES ARE[ NOT] ALLOWED        ⎭

    WITHIN realmname-1[,realmname-2,... AREA-ID IS bezeichner-2]

    [SEARCH KEY IS feldname-3,...USING ⎧CALC[ hashroutine]⎫[ NAME IS name]
                                       ⎩INDEX             ⎭

        DUPLICATES ARE[ NOT] ALLOWED]...,

    {[stufennummer ]satzelementname

          ⎧                ⎧maskenzeichenkette                          ⎫      ⎫
          ⎪PICTURE IS      ⎨                                            ⎬      ⎪
          ⎪                ⎩LX(ganzzahl-1) DEPENDING ON feldname-4      ⎭      ⎪
          ⎪                ⎧                ⎧BINARY[ ⎧15⎫]                ⎫      ⎪
          ⎪                ⎪FIXED REAL      ⎨        ⎩31⎭                 ⎬      ⎪
        [ ⎨                ⎩                ⎩DECIMAL[ ganzzahl-2[,ganzzahl-3]]⎭      ⎬]
          ⎪TYPE IS         ⎧CHARACTER[ ganzzahl-4[ DEPENDING ON feldname-5]]⎫      ⎪
          ⎪                ⎪DATABASE-KEY                                    ⎪      ⎪
          ⎪                ⎩DATABASE-KEY-LONG                               ⎭      ⎪
          ⎩                                                                        ⎭

        [OCCURS ganzzahl-5 TIMES],}...
```

Sie benennen eine Satzart und

- bestimmen die Verteilung ihrer Sätze auf Realms,
- bestimmen ggf. die Reihenfolge der Sätze zur sequenziellen Verarbeitung,
- legen ggf. zusätzliche Zugriffspfade für Direktzugriff über Primär- und Sekundärschlüssel an,
- definieren alle Satzelemente, die der Satzart angehören sollen.

> **i** In einer Datenbank mit 2048 byte Seitenlänge können Sie maximal 253 Satzarten definieren.
> In einer Datenbank mit 4000 byte oder 8096 byte Seitenlänge können Sie maximal 32 766 Satzarten definieren.

Die Klauseln des Satz-Eintrags sind nachfolgend erläutert.

```
RECORD NAME IS satzname
```

Sie benennen die Satzart.

```
                      ⎧⎧DIRECT     ⎫ ⎧feldname-1 ⎧IN⎫ satzname⎫⎫
                      ⎪⎨           ⎬ ⎨           ⎨  ⎬         ⎬⎪
                      ⎪⎩DIRECT-LONG⎭ ⎪           ⎩OF⎭         ⎪⎪
[LOCATION MODE IS     ⎨              ⎩bezeichner              ⎭⎬]
                      ⎪CALC[ hashroutine] USING feldname-2,...  ⎪
                      ⎩     DUPLICATES ARE[ NOT] ALLOWED        ⎭
```

---

feldname-1
: muss ein Database-Key-Feld bezeichnen.
Wenn Sie LOCATION MODE IS DIRECT angeben, müssen Sie *feldname-1* als DATABASE-KEY-Feld definieren.
Wenn Sie LOCATION MODE IS DIRECT-LONG angeben, müssen Sie *feldname-1* als DATABASE-KEY-LONG-Feld definieren

feldname-2
: muss ein Feld fester Länge bezeichnen, das dieser Satzart angehört.

Mit LOCATION MODE IS DIRECT/DIRECT-LONG schaffen Sie die Möglichkeit, den Database Key eines zu speichernden Satzes selbst zu vergeben und die Reihenfolge bei sequentieller Verarbeitung selbst zu bestimmen. Informationen zur Vergabe eines Database Keys beim Speichern eines Satzes finden sie im Handbuch „Anwendungen programmieren“ im Nachschageteil der COBOL-DML unter „STORE, Database-Key-Wert zuordnen“.

Mit LOCATION MODE IS CALC legen Sie einen Primärschlüssel zum Zwecke des Direktzugriffs auf einen bestimmten Satz oder eine Menge von Sätzen mit gleichen Schlüsselwerten an.

Im Allgemeinen wird ein direkter CALC angelegt. In Sonderfällen (z.B. für eine Membersatzart einer Liste) wird ein indirekter CALC angelegt.

```
WITHIN realmname-1[,realmname-2,... AREA-ID IS bezeichner]
```

realmname-1,-2,...
dürfen keinen Temporären Realm nennen.

Sie bestimmen die Verteilung der Sätze dieser Satzart auf Realms.

Bei verteilbaren Listen legen Sie hier mit dem ersten angegebenen Realm die Lage des Tabellenteils der Liste (Tabellenrealm) und evtl. eines indirekten CALCs implizit fest, sofern diese nicht explizit mit der DETACHED WITHIN-Klausel in der SSL bestimmt wird.

```
[SEARCH KEY IS feldname,... USING {CALC[ hashroutine]}[ NAME IS name]
                                  {INDEX              }
       DUPLICATES ARE[ NOT] ALLOWED]...,
```

feldname
muss ein Feld fester Länge bezeichnen, das dieser Satzart angehört.

name bezeichnet Tabellen für SEARCH-Keys; in den SSL-Anweisungen wird darauf Bezug genommen.

Sie legen zusätzliche Direktzugriffspfade über Sekundärschlüssel an und benennen die SEARCH-KEY-Tabelle bzw. den Hashbereich, um ihn mit der SSL ansprechen zu können.

```
[stufennummer ]satzelementname
```

stufennummer
muss eine Ganzzahl zwischen 1 und 99 sein.
Standardwert ist 1.

Sie benennen ein Satzelement und legen fest, ob es zu einer Wiederholungsgruppe gehört.

Die Gesamtlänge aller Satzelemente innerhalb einer Satzart darf die maximale Satzlänge nicht überschreiten.
Die maximale Satzlänge beträgt:

- 2020 byte in einer Datenbank mit 2048 byte Seitenlänge (2-Kbyte-Format)
- 3968 byte in einer Datenbank mit 4000 byte Seitenlänge (4-Kbyte-Format)
- 8064 byte in einer Datenbank mit 8096 byte Seitenlänge (8-Kbyte-Format)

Abhängig von der Länge der SCD kann die maximale Satzlänge geringfügig kleiner sein.

```
PICTURE IS {maskenzeichenkette                    }
           {LX(ganzzahl-1) DEPENDING ON feldname  }
```

maskenzeichenkette

darf aus folgenden Symbolen bestehen:

Symbol	Syntaxregel
S	geben Sie höchstens einmal an der ersten Stelle der Maskenzeichenkette an.
X	Mindestens eines dieser Zeichen müssen Sie angeben. Sie können es mehrfach angeben oder ein Wiederholungssymbol anfügen. 9 darf nicht links von einem A oder X stehen. N darf nicht mit anderen Zeichen kombiniert werden.
A	
9	
N	
V	kann nicht zusammen mit X, N oder A verwendet werden.
P	kann nicht zusammen mit X, N oder A verwendet werden. Sie können es mehrfach angeben oder ein Wiederholungssymbol anfügen. P dürfen Sie nicht gleichzeitig links und rechts von 9 angeben.
(*ganzzahl*)	können Sie an X, N, A, 9 oder P anfügen.

Tabelle 24: Maskenzeichenkette

Die Maskenzeichenkette darf eine Feldlänge von maximal 255 Byte definieren. Dies entspricht in der Regel der Anzahl der Zeichen. Das Symbol N dürfen Sie jedoch maximal 127 Mal wiederholen , da in diesem Fall ein Zeichen 2 Byte belegt. Das Symbol 9 dürfen Sie höchstens 18 Mal wiederholen.

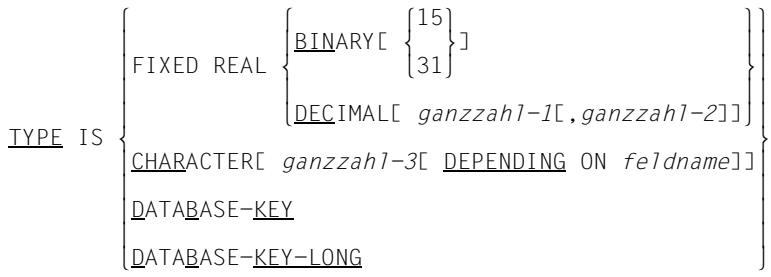
ganzzahl-1

muss > 0 sein. Der Maximalwert ist abhängig vom Satzaufbau.

feldname

muss ein Feld bezeichnen, das Sie unmittelbar vorausgehend mit TYPE IS BINARY 15 definiert haben.

Mit PICTURE IS definieren Sie ungepackte numerische Felder, alphanumerische Felder fester oder variabler Länge oder nationale Felder.



BINARY
: Wenn Sie keine Zahl angeben, wird standardmäßig 15 angenommen.

ganzzahl-1
: muss zwischen 1 und 18 liegen.
Standardwert ist 18

ganzzahl-2
: darf nicht größer als 18 und nicht kleiner als {ganzzahl-1} -18 sein.
Standardwert ist 0.

ganzzahl-3
: wenn Sie nicht DEPENDING ON verwenden, muss *ganzzahl-3* zwischen 1 und 255 liegen. Im anderen Fall ist der Maximalwert abhängig vom Satzaufbau.

feldname
: muss ein Feld bezeichnen, das Sie unmittelbar vorausgehend mit TYPE IS BINARY 15 definiert haben.

Hiermit definieren Sie

- gepackte numerische Felder,
- binäre Felder,
- alphanumerische Felder fester oder variabler Länge oder
- Database-Key-Felder.

[OCCURS *ganzzahl* TIMES]

ganzzahl
: muss > 0 sein. Der Maximalwert ist abhängig vom Satzaufbau.

Sie definieren den Wiederholungsfaktor für einen Vektor oder eine Wiederholungsgruppe.

9.1.4 Der Set-Eintrag

```
SET NAME IS setname
    [SET IS DYNAMIC]
    ORDER IS { LAST
             | FIRST
             | NEXT
             | PRIOR
             | IMMATERIAL
             | SORTED[ INDEXED[ NAME IS name]]
               BY { DATABASE-KEY
                  | DEFINED KEYS DUPLICATES ARE[ NOT] ALLOWED } }
    OWNER IS { satzname
             | SYSTEM }.
[MEMBER IS satzname { MANDATORY } { AUTOMATIC }
                    { OPTIONAL  } { MANUAL    }
    [{ ASCENDING  } KEY IS feldname-1,...]
     { DESCENDING }
    [SEARCH KEY IS feldname-2,... USING { CALC[ hashroutine] }[NAME IS name]
                                        { INDEX              }
        DUPLICATES ARE[ NOT] ALLOWED]...
    [SET OCCURRENCE SELECTION IS
          THRU { CURRENT OF SET
               | LOCATION MODE OF OWNER[ ALIAS FOR { feldname-3   }]]].
                                                   { bezeichner-1 }
                                  IS bezeichner-2]... }
```

Sie benennen einen Set und

- erklären den Set ggf. zum dynamischen Set,
- legen die Reihenfolge der Membersätze innerhalb der Set-Occurrences für eine sequenzielle Verarbeitung fest,
- legen ggf. zusätzliche Zugriffspfade über Primär- und Sekundärschlüssel an,
- bestimmen eine Satzart zur Ownersatzart des Set,
- bestimmen ggf. eine Satzart zur Membersatzart des Set und definieren die Art der Set-Mitgliedschaft der Membersätze und
- bestimmen ggf. die Auswahlmethode für die Set-Occurrences des Set.

Pro Datenbank können Sie maximal 32 766 Sets definieren.

Pro Satzart, die Owner eines Sets ist, dürfen Sie in diesen Sets in Summe maximal 255 Tabellen erzeugen; eine Tabelle wird angelegt, wenn der Set-Mode Pointer-Array oder List oder Chain sorted indexed ist, ebenso für jeden Sekundärschlüssel in diesen Sets.

Unabhängig davon dürfen Sie maximal 255 Sekundärschlüssel auf Satzartebene pro Satzart sowie auf Setebene pro singulärem Set definieren, wobei Hashverfahren nicht mitzählen.

Die Klauseln des Set-Eintrags sind nachfolgend erläutert.

SET NAME IS *setname*

Sie benennen den Set.

[SET IS DYNAMIC]

Sie erklären den Set zum dynamischen Set.

```
                ⎧LAST                                                  ⎫
                ⎪                                                      ⎪
                ⎪FIRST                                                 ⎪
                ⎪                                                      ⎪
                ⎪NEXT                                                  ⎪
                ⎪                                                      ⎪
                ⎪PRIOR                                                 ⎪
ORDER IS        ⎨                                                      ⎬
                ⎪IMMATERIAL                                            ⎪
                ⎪                                                      ⎪
                ⎪SORTED[ INDEXED[ NAME IS name]]                       ⎪
                ⎪                                                      ⎪
                ⎪     ⎧DATABASE-KEY                                  ⎫ ⎪
                ⎪  BY ⎨                                              ⎬ ⎪
                ⎩     ⎩DEFINED KEYS DUPLICATES ARE[ NOT] ALLOWED     ⎭ ⎭
```

Sie legen

- die Reihenfolge der Sätze innerhalb der Set-Occurrences für eine sequenzielle Verarbeitung fest.
- einen zusätzlichen Direktzugriffspfad über den Primärschlüssel an.

```
OWNER IS ⎧satzname⎫.
         ⎩SYSTEM  ⎭
```

Sie bestimmen eine von Ihnen definierte Satzart oder die symbolische Satzart SYSTEM zur Ownersatzart des Set.

```
                                ⎧MANDATORY⎫ ⎧AUTOMATIC⎫
MEMBER IS satzname              ⎨         ⎬ ⎨         ⎬
                                ⎩OPTIONAL ⎭ ⎩MANUAL   ⎭
```

Die gesamte Beschreibung der Membersatzart entfällt im dynamischen Set.
Sonst bestimmen Sie hiermit eine Satzart zur Membersatzart und definieren die Art der Set-Mitgliedschaft der Membersätze.

```
[{ASCENDING }
 {          } KEY IS feldname,...]
 {DESCENDING}
```

feldname,...
muss ein Feld fester Länge bezeichnen, das dieser Satzart angehört.

Hiermit definieren Sie ein Feld oder eine Kombination von Feldern der Membersatzart zum Sortierschlüssel, nach dessen Werten die Membersätze innerhalb der Set-Occurrence auf- oder absteigend sortiert werden.

```
[SEARCH KEY IS feldname,... USING {CALC[ hashroutine]}[ NAME IS name]
                                  {                  }
                                  {INDEX             }
    DUPLICATES ARE[ NOT] ALLOWED]...
```

feldname,...
muss ein Feld fester Länge bezeichnen, das dieser Satzart angehört.

name bezeichnet den Namen der Tabelle; in den SSL-Anweisungen wird darauf Bezug genommen.

Sie legen zusätzliche Direktzugriffspfade über Sekundärschlüssel an und benennen die SEARCH-Key-Tabelle bzw. den Hashbereich, um ihn mit der SSL ansprechen zu können.

Die Angabe SEARCH KEY ... USING CALC ist nur beim SYSTEM-Set zulässig.

```
[SET OCCURRENCE SELECTION IS

          ┌CURRENT OF SET                                          ┐
          │                                                        │
          │                                          ┌feldname     ┐ │
     THRU ┤LOCATION MODE OF OWNER[ ALIAS FOR ┤                     ├]
          │                                          └bezeichner-1 ┘ │
          │                                                        │
          └                              IS bezeichner-2]...       ┘
```

feldname

muss ein Feld bezeichnen, das Sie in der LOCATION-MODE-Klausel der Ownersatzart verwendet haben.

bezeichner-1

muss ein Bezeichner sein, den Sie in der LOCATION MODE-Klausel der Ownersatzart verwendet haben.

bezeichner-2

Mit *bezeichner-2* vergeben Sie den Namen für das zusätzlich zu erzeugende Feld. UDS/SQL richtet dieses Feld automatisch mit demselben Feldtyp und derselben Feldlänge ein wie das Feld *feldname* bzw. *bezeichner-1*.

Diese Klausel müssen Sie genau dann angeben, wenn der Set kein SYSTEM-Set ist. Sie bestimmen damit die Auswahlmethode für die Set-Occurrences des Set.

9.2 Syntax der SSL

```
Schema-Eintrag  STORAGE-Klausel.

                ⎧[RECORD NAME-Klausel
                ⎪[DATABASE-KEY-TRANSLATION-TABLE-Klausel]
                ⎪[Satz-POPULATION-Klausel]
Satz-Eintrag    ⎨[PLACEMENT-OPTIMIZATION-Klausel]
                ⎪[INDEX-Klausel]
                ⎩[COMPRESSION-Klausel]].

                ⎧[SET NAME-Klausel
                ⎪[Set-POPULATION-Klausel]
Set-Eintrag     ⎨[MODE-Klausel]
                ⎪[INDEX-Klausel]
                ⎩[PHYSICALLY LINKED-Klausel]].
```

Bild 63: Aufbau der SSL

Die Beschreibung der physischen Speicherstruktur ist wahlfrei. Wenn Sie darauf verzichten, benutzt UDS/SQL voreingestellte Standardwerte, die bei nachfolgenden Erläuterungen der einzelnen Syntaxelemente jeweils angegeben sind.

Im anderen Fall beginnen Sie die Beschreibung immer mit der STORAGE-Klausel. Die Abfolge der Satz- und Set-Beschreibungen ist beliebig.
Alle Namen, die Sie in der Speicherstrukturbeschreibung benutzen, müssen Sie zuvor mit der Schema-DDL definiert haben.

9.2.1 Der Schema-Eintrag

```
STORAGE STRUCTURE OF SCHEMA schemaname.
```

Sie geben den Namen des Schemas an, auf das sich die Beschreibung der Speicherstruktur beziehen soll.

9.2.2 Der Satz-Eintrag

```
RECORD NAME IS satzname
    [DATABASE-KEY-TRANSLATION-TABLE[ IS ganzzahl-1][ WITHIN realmname-1]]
    [POPULATION IS {ganzzahl-2 WITHIN realmname-2},...]
    [PLACEMENT OPTIMIZATION FOR SET setname]
    [INDEX NAME IS name
        [PLACING IS WITHIN realmname-3]
                  ⎧DATABASE-KEY-LIST                                  ⎫
        [TYPE IS  ⎨REPEATED-KEY                                       ⎬]]...
                  ⎩  [DYNAMIC REORGANIZATION SPANS ganzzahl-3 PAGES]  ⎭
    [COMPRESSION FOR ALL ITEMS].
```

Im Satz-Eintrag nennen Sie den Namen der Satzart, auf die sich die Beschreibung der Speicherstruktur bezieht und machen Angaben

- zur Größe und physischen Lage der DBTT und zur Größe der Hashbereiche für Satz-SEARCH-Keys,
- zur Anzahl der Sätze der Satzart bzw. zur Größe des Hashbereiches für den Primärschlüssel innerhalb bestimmter Realms,
- zur physischen Lage der Sätze innerhalb eines Realms, falls die Satzart Member in einem Set ist,
- zur physischen Lage, zum Typ und zum Reorganisationsaufwand der Satz-SEARCH-Key-Tabellen bzw. zur physischen Lage der Hashbereiche für Satz-SEARCH-Keys und
- zur komprimierten Speicherung.

Die Klauseln des Satz-Eintrages sind nachfolgend erläutert.

```
RECORD NAME IS satzname
```

Sie geben den Namen der Satzart an, auf die sich der Satz-Eintrag bezieht.

```
[DATABASE-KEY-TRANSLATION-TABLE[ IS ganzzahl][ WITHIN realmname]]
```

ganzzahl
: muss > 0 sein. Wenn Sie keine Angabe machen, legt UDS/SQL die DBTT und den Hashbereich eines Satz-SEARCH-Key in der Größe einer Seite an.

realmname
: darf keinen Temporären Realm bezeichnen. Wenn Sie keine Angabe machen, kommt die DBTT im ersten Realm der DDL-WITHIN-Klausel dieser Satzart zu liegen.

Sie geben die Größe und Lage der DBTT und gleichzeitig die Größe der Hashbereiche für Satz-SEARCH-Keys an.

```
[POPULATION IS {ganzzahl WITHIN realmname},...]
```

ganzzahl
: muss > 0 sein.

realmname
: muss ein Realm-Name aus der DDL-WITHIN-Klausel dieser Satzart sein.
Sie müssen alle Realm-Namen der DDL-WITHIN-KLausel dieser Satzart aufzählen.

Hier geben Sie die Größe und die Lage von Hashbereichen für den Primärschlüssel (LOCATION MODE IS CALC) an. Die Angabe dient UDS/SQL außerdem zur Abschätzung der Realm-Größen.

Wenn Sie die Klausel nicht verwenden, reserviert UDS/SQL in jedem Realm der WITHIN-Klausel eine Seite für den Hashbereich, wenn die Satzart nicht Membersatzart einer verteilbaren Liste ist.

Bei einer verteilbaren Liste wird der Hashbereich nur in einem Realm angelegt. Die Größe des Hashbereichs orientiert sich an der Summe der für die verschiedenen Realms angegebenen Werte für die POPULATION.

```
[PLACEMENT OPTIMIZATION FOR SET setname]
```

setname
darf keinen SYSTEM-Set bezeichnen.

Hiermit erreichen Sie eine Bündelung der Sätze bei dem Ownersatz des Set *setname*.

Die Satzart muss AUTOMATIC-Member des Set *setname* sein.

Jeder Realm, der in der DDL-WITHIN-Klausel dieser Satzart genannt ist, muss auch in der WITHIN-Klausel der Ownersatzart des Set *setname* angegeben sein.

ganzzahl-1 in der Set-POPULATION-Klausel des Set *setname* muss > 0 sein.

Wenn Sie für die Satzart LOCATION MODE IS CALC definiert haben, entsteht ein indirekter Hashbereich.

In folgenden Fällen ist die Klausel wirkungslos:

- Die Satzart ist Member in einem Set, den Sie mit MODE IS LIST definiert haben.
- Die Ownersatzart des Set *setname* ist Member in einem Set, den Sie mit MODE IS LIST definiert haben.
- Sie haben für die Ownersatzart des Set *setname* LOCATION MODE IS CALC oder PLACEMENT OPTIMIZATION definiert.

```
[INDEX NAME IS name

    [PLACING IS WITHIN realmname]

                ⎧DATABASE-KEY-LIST                                  ⎫
                ⎪                                                   ⎪
    [TYPE IS    ⎨REPEATED-KEY                                       ⎬]]...
                ⎪                                                   ⎪
                ⎩  [DYNAMIC REORGANIZATION SPANS ganzzahl PAGES]     ⎭
```

name müssen Sie mit der Schema-DDL für eine Satz-SEARCH-Key-Tabelle oder einen Hashbereich dieser Satzart vereinbart haben.

realmname
darf keinen Temporären Realm nennen. Wenn Sie keine Angabe machen, platziert UDS/SQL die Satz-SEARCH-Key-Tabelle bzw. den Hashbereich in den ersten Realm der DDL-WITHIN-Klausel dieser Satzart.

ganzzahl
muss eine Zahl zwischen 1 und 20 sein.
Standardwert ist 2.

Sie nennen den Namen der Satz-SEARCH-Key-Tabelle bzw. des Hashbereichs auf den sich die Beschreibung beziehen soll und bestimmen

– für eine Satz-SEARCH-Key-Tabelle die physische Lage, den Typ und den Reorganisationsaufwand,
– für einen Hashbereich die physische Lage.

i Wenn sich die Beschreibung auf einen Hashbereich bezieht, ist nur die PLACING-Angabe zulässig.

```
[COMPRESSION FOR ALL ITEMS]
```

Sie veranlassen UDS/SQL, die Sätze komprimiert zu speichern, wenn sie an der DML-Schnittstelle in komprimierter Form zur Verfügung gestellt werden.

Wenn die Satzart mit LOCATION MODE IS CALC definiert ist, entstehen durch die Komprimierung indirekte CALC-Seiten.

Komprimierung ist nicht erlaubt, wenn die Satzart ein Feld variabler Länge enthält oder Member in einem Set ist, der mit MODE IS LIST definiert ist.

9.2.3 Der Set-Eintrag

```
SET NAME IS setname

    [POPULATION IS ganzzahl-1[ INCREASE IS ganzzahl-2]]

                ⎧CHAIN[ LINKED TO PRIOR]                          ⎫
                ⎪                                                 ⎪
    [MODE IS    ⎨⎧POINTER-ARRAY⎫ ⎧ATTACHED TO OWNER              ⎫⎬]
                ⎪⎨             ⎬ ⎨                               ⎬⎪
                ⎩⎩LIST         ⎭ ⎩DETACHED[ WITHIN realmname-1]  ⎭⎭
                                   [ WITH PHYSICAL LINK]

    [DYNAMIC REORGANIZATION SPANS ganzzahl-3 PAGES]

    [INDEX NAME IS name

        [PLACING IS ⎧ATTACHED TO OWNER             ⎫
                    ⎩DETACHED[ WITHIN realmname-2]  ⎭]

                ⎧DATABASE-KEY-LIST                                  ⎫
                ⎪                                                   ⎪
    [TYPE IS    ⎨REPEATED-KEY                                       ⎬]]...
                ⎪                                                   ⎪
                ⎩  [DYNAMIC REORGANIZATION SPANS ganzzahl-4 PAGES]   ⎭

    [MEMBER IS PHYSICALLY LINKED TO OWNER].
```

Im Set-Eintrag nennen Sie den Namen des Set, auf den sich die Beschreibung der Speicherstruktur bezieht und machen Angaben

- zum durchschnittlichen Umfang der Set-Occurrences,
- zur Verknüpfung der Sätze innerhalb der Set-Occurrences,
- zur Platzierung und zum Reorganisationsaufwand von Adresslisten, Listen, Sort-Key-Tabellen und Set-SEARCH-Key-Tabellen,
- zum Typ von Set-SEARCH-Key-Tabellen,
- für einen zusätzlichen Zeiger vom Member auf seinen Owner.

Die Klauseln des Set-Eintrags sind nachfolgend erläutert.

```
SET NAME IS setname
```

Hier geben Sie den Namen des Set an, auf den sich der Set-Eintrag bezieht.

```
[POPULATION IS ganzzahl-1[ INCREASE IS ganzzahl-2]]
```

ganzzahl-1
muss ≥ 0 sein. Der Standardwert ist 0.

ganzzahl-2
muss > 0 sein. Der Standardwert ist 1.

Sie geben die durchschnittliche Anzahl von Membersätzen der Set-Occurrences an, die Sie beim Urladen der Datenbank bzw. bei späteren Erweiterungen der Set-Occurrences erwarten.

```
                ⎧CHAIN[ LINKED TO PRIOR]                          ⎫
                ⎪                                                 ⎪
                ⎪⎧POINTER-ARRAY⎫ ⎧ATTACHED TO OWNER               ⎫⎪
[MODE IS        ⎨⎪             ⎬ ⎨                               ⎬⎬]
                ⎪⎩             ⎭ ⎩DETACHED[ WITHIN realmname]    ⎭⎪
                ⎪                                                 ⎪
                ⎩⎧LIST         ⎫ ⎧        [WITH PHYSICAL LINK]   ⎫⎭
                 ⎩             ⎭ ⎩                               ⎭

[DYNAMIC REORGANIZATION SPANS ganzzahl PAGES]
```

CHAIN [LINKED TO PRIOR]
ist die einzige erlaubte Angabe, wenn Sie den Set mit der Schema-DDL als ORDER IS SORTED (ohne INDEXED) definiert haben.

POINTER-ARRAY DETACHED WITHIN *realmname*
ist die einzige erlaubte Angabe, wenn der Set dynamisch ist.

LIST dürfen Sie nur angeben, wenn die folgenden Bedingungen erfüllt sind:

- Die Set-Mitgliedschaft der Membersatzart ist als MANDATORY AUTOMATIC definiert.
- die Membersätze (einschl. Zeigern, siehe Seite 217, SCD) sind nicht länger als
  - 993 byte bei Datenbanken mit 2048 byte Seitenlänge,
  - 1963 byte bei Datenbanken mit 4000 byte Seitenlänge,
  - 4011 byte bei Datenbanken mit 8096 byte Seitenlänge.

ATTACHED

dürfen Sie nur angeben, wenn der Set kein SYSTEM-Set ist.
Bei MODE IS LIST ist diese Angabe nicht erlaubt, wenn die Ownersatzart Membersatzart einer verteilbaren Liste ist.

LIST DETACHED (ohne WITHIN)

- Falls die DDL-WITHIN-Klausel der Membersatzart nur einen Realm enthält, wird die Liste bei DETACHED (ohne WITHIN) in diesem Realm abgelegt.
- Falls die DDL-WITHIN-Klausel der Membersatzart mehr als einen Realm enthält, wird wie folgt verfahren:
  - Falls der Set kein SYSTEM-Set ist, müssen die angegebenen Realms in der DDL-WITHIN-Klausel von Owner- und Membersatzart übereinstimmen. Die Lage der Liste wird dann durch die Lage des Owners bestimmt. Die Ownersatzart darf nicht Membersatzart einer verteilbaren Liste sein.
  - Falls der Set ein SYSTEM-Set ist, liegt bei ORDER SORTED INDEXED eine verteilbare Liste vor. Der erste Realm der DDL-WITHIN-Klausel der Membersatzart bestimmt die Lage des Index und die Lage eines möglicherweise existierenden indirekten Hashbereiches. Der erste Realm der DDL-WITHIN-Klausel der Membersatzart bestimmt auch die Lage einer nicht sortierten Liste.
- Bei verteilbaren Listen wird die Lage des Tabellenteils (Seiten mit Stufe > 0) durch den ersten in der DDL-WITHIN-Klausel der Membersatzart angegebenen Realm bestimmt. In diesem Realm liegt dann auch ein evtl. deklarierter indirekter Hashbereich für die gesamte Liste.

LIST DETACHED WITHIN

bestimmt den Realm, in dem die Liste abgelegt wird.
Bei verteilbaren Listen bestimmt die DETACHED WITHIN-Klausel die Lage des Tabellenteils (Seiten mit Stufe > 0) bzw. des indirekten Hashbereichs der gesamten Liste.

realmname

muss genau dann einen Temporären Realm bezeichnen, wenn der Set dynamisch ist. Bei LIST muss er aus der DDL-WITHIN-Klausel der Membersatzart stammen.

ganzzahl

muss eine Zahl zwischen 1 und 20 sein.
Standardwert ist 2.

Sie bestimmen die Verknüpfung der Sätze innerhalb der Set-Occurrences.
Falls Sie eine Adressliste, Liste oder Sort-Key-Tabelle anlegen, bestimmen Sie auch den Reorganisationsaufwand für diese Tabellen.

Wenn Sie die Klausel nicht anwenden, gelten folgende Standardwerte:

- In dynamischen Sets: POINTER-ARRAY DETACHED WITHIN *realmname*
- Bei ORDER IS SORTED INDEXED: POINTER-ARRAY DETACHED
- Bei ORDER IS LAST/FIRST/NEXT/PRIOR/SORTED: CHAIN
- Bei ORDER IS IMMATERIAL, wenn der Set nicht dynamisch ist: CHAIN

```
[INDEX NAME IS name

     [PLACING IS {ATTACHED TO OWNER                    }]
                 {DETACHED[ WITHIN realmname]          }

                 {DATABASE-KEY-LIST                                  }
     [TYPE IS    {REPEATED-KEY                                       }]]...
                 {    [DYNAMIC REORGANIZATION SPANS ganzzahl PAGES]   }
```

---

name muss der Name einer Tabelle oder eines Hashbereichs sein, den Sie in der Schema-DDL vergeben haben.

ATTACHED
dürfen Sie nur angeben, wenn der Set kein SYSTEM-Set ist.

realmname
darf keinen Temporären Realm bezeichnen. Wenn Sie keine Angabe machen, wählt UDS/SQL den Standardwert gemäß Seite 159, Bild 40.

ganzzahl
muss eine Zahl zwischen 1 und 20 sein. Standardwert ist 2.

Wenn Sie mit der Schema-DDL eine Tabelle dieses Set benannt haben, können Sie die Tabelle hier ansprechen, um die Lage, den Typ und den Reorganisationsaufwand für die Tabelle festzulegen.
Wenn Sie einen SEARCH-Key zur Speicherung in einen Hashbereich definiert haben, können Sie den Realm bestimmen, in dem der Hashbereich liegen soll.

Standardwerte sind:

- für PLACING: DETACHED
- für TYPE: REPEATED-KEY

MEMBER IS PHYSICALLY LINKED TO OWNER

Hiermit legen Sie in jedem Membersatz dieses Set einen zusätzlichen Zeiger zu ihrem zugehörigen Owner an: Die SCD eines Membersatzes enthält zusätzlich den Probable Position Pointer (PPP) des zugehörigen Ownersatzes.

9.3 Syntax der Subschema-DDL

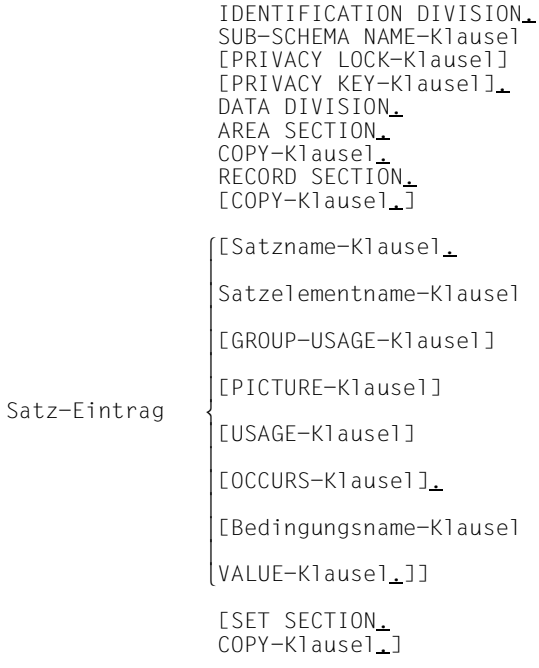


Bild 64: Aufbau der Subschema-DDL

Die im Bild 64 gezeigte Abfolge von Klauseln müssen Sie mit folgenden Ausnahmen einhalten:

– Die Reihenfolge von PICTURE- und USAGE-Klausel ist beliebig.

– Eine COPY-Klausel in der RECORD SECTION darf auch auf einen Satz-Eintrag folgen.

Namen, die Sie in der Definition des Subschemas benutzen, müssen Sie zuvor mit der Schema-DDL vereinbart haben.
Die Ausnahme sind Namen für Datengruppen und Bedingungen, die Sie im Subschema neu definieren.

9.3.1 IDENTIFICATION DIVISION

```
IDENTIFICATION DIVISION.
    SUB-SCHEMA NAME IS subschemaname OF SCHEMA NAME schemaname
    [PRIVACY LOCK FOR COMPILE IS literal-1[ OR literal-2]]
    [PRIVACY KEY FOR COPY IS literal-3].
```

literal-1,-2
darf maximal 10 Zeichen umfassen.

literal-3
muss ein Kennwort sein, das im Schema-Eintrag der Schema-DDL vergeben wurde.

Hiermit benennen Sie das Subschema und

- geben an, aus welchem Schema Sie das Subschema entnehmen wollen,
- vereinbaren Kennwörter zum Schutz gegen unbefugte Übersetzung eines DML-Programms mit diesem Subschema und
- nennen ggf. eines der Kennwörter, die das Schema vor unbefugter Entnahme eines Subschemas schützen.

9.3.2 AREA SECTION

```
DATA DIVISION.
AREA SECTION.

    {COPY ALL AREAS.                    }
    {                                   }
    {{COPY realmname,...,}...           }
```

Hiermit übernehmen Sie alle Realms oder eine Auswahl von Realms aus dem Schema in das Subschema.

9.3.3 RECORD SECTION

```
RECORD SECTION.

     [{COPY ALL RECORDS.                  }]
      {{COPY satzname-1,...}.,...}

[01  satzname-2.
     {stufennummer satzelementname[ PICTURE IS maskenzeichenkette]

                                    [ GROUP-USAGE IS NATIONAL]

                                                {DISPLAY                }
                                                {COMPUTATIONAL-3        }
                                                {COMPUTATIONAL          }
                                    [ USAGE IS  {NATIONAL               }]
                                                {DATABASE-KEY           }
                                                {DATABASE-KEY-LONG      }

                                    [ OCCURS ganzzahl TIMES].}...

      [88 bedingungsname

           {VALUE IS  } {literal-1[ THROUGH literal-2]},... .]...]...
           {VALUES ARE}
```

satzname-2
: darf nicht mit *satzname-1* übereinstimmen und nicht gleichzeitig mit COPY ALL RECORDS verwendet werden.

stufennummer
: muss eine Ganzzahl zwischen 02 und 49 sein.

maskenzeichenkette
: siehe Seite 237, Tabelle 24

GROUP-USAGE
: Wenn die GROUP-USAGE-Klausel angegeben ist, müssen alle untergeordneten Satzelemente vom Typ NATIONAL sein.

USAGE

Wenn Sie keine Angabe machen, wird standardmäßig DISPLAY angenommen.

Ausnahme: Wenn die PICTURE-Klausel das Symbol N enthält, wird bei fehlender USAGE-Klausel NATIONAL angenommen.

literal-1

muss kleiner sein als *literal-2*.

Sie entscheiden sich, ob Sie alle im Schema vorhandenen Satzarten komplett oder nur eine Auswahl von Satzarten bzw. Feldern in das Subschema übernehmen wollen. Im letzteren Fall nennen Sie die Satzarten, die Sie komplett oder nur teilweise übernehmen wollen. Bei Sätzen, die Sie nur teilweise übernehmen, nennen Sie alle Satzelemente, die Sie übernehmen wollen.
Zusätzlich können Sie Datengruppen und Bedingungen definieren.

9.3.4 SET SECTION

```
SET SECTION.

    {COPY ALL SETS.                    }
    {{COPY setname,...,}...            }
```

Hiermit übernehmen Sie alle Sets oder eine Auswahl von Sets aus dem Schema.

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 *Primärschlüssel*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.

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.*

U

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.

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

Literatur

Die Handbücher finden Sie im Internet unter *http://manuals.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)
Aufbauen und Umstrukturieren
Benutzerhandbuch

UDS/SQL (BS2000)
Datenbankbetrieb
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

Stichwörter

C

D

E

F