Deutsch

FUJITSU Software BS2000

UDS/SQL V2.8

Anwendungen programmieren

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 15.

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

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

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

Beispiele

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

Allgemeines	E = zum Einstieg
Schema-DDL	D = zur Detailinformation

SSL D = zur Detailinformation

Subschema-DDL L = zum Lernen der Funktionen

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

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

Inhalt der fünf Haupthandbücher	Aufgaben des Anwenders							
	Entwerfen und Definieren	COBOL/ CALL-DML Programm.	SQL-Program-mieren	Aufbauen und Umstrukt.	Verwalten und Bedienen	Arbeiten mit openUTM	Arbeiten mit IQS	Arbeiten mit UDS-D
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	–	–	–	–
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	–	–	–	–	–	–

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 Aufbauen und Umstrukturieren								
Einleitung	–	–	–	E	–	E	E	–
Überblick	–	–	–	E	E	–	–	–
Datenbank aufbauen	–	–	–	L	–	–	–	–
Zugriffsberechtigungen festlegen	–	–	–	L	–	–	–	–
Daten speichern und entladen	D	–	–	L	–	D	–	–
Datenbank umstrukturieren	D	–	–	L	–	–	–	–
Datenbankobjekte umbenennen	D	–	–	L	–	–	–	–
Datenbank umstellen	D	–	–	L	–	–	–	–
Handbuch UDS/SQL Datenbankbetrieb								
Einleitung	–	–	–	–	E	E	E	–
Der Database Handler	–	–	–	–	L	–	–	D
Ladeparameter des DBH	–	–	–	–	L	–	–	D
Administration	–	–	–	–	L	–	–	D
Hochverfügbarkeit	–	–	–	–	E	–	–	–
Ressourcen-Erweiterung und Umorganisation im laufenden Betrieb	D	–	–	–	E	–	–	–
Datenbank sichern und wiederherstellen im Fehlerfall	D	–	–	D	L	D	–	D
Leistungsoptimierung	–	–	–	–	D	–	–	D
Nutzung der BS2000-Funktionalität	–	–	–	–	D	–	–	–
Der SQL-Vorgang	–	–	–	–	L	–	–	–
UDSMON	–	–	–	–	D	–	–	–
Einsatz von IQS	–	–	–	L	D	–	D	–
Einsatz von UDS-D	D	D	–	D	D	D	–	D
Funktionscodes der DML-Anweisungen	–	D	–	–	D	–	–	–

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
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 den Programmierer von Datenbankanwendungen bestimmt, der die Aufgabe hat, Problemstellungen für eine UDS/SQL-Datenbank in Anweisungen umzusetzen. Er muss dabei die Struktur der Datenbank berücksichtigen und gegebenenfalls darauf Einfluss nehmen. Das Subschema, das er für seine Anwendung benötigt, muss alle notwendigen Informationen enthalten.

1.3 Konzept des Handbuchs

Was enthält dieses Handbuch?

Einführend sind im Handbuch Sprachkonzept und Funktionsumfang der Datenbehandlungssprache DML beschrieben.

In den darauf folgenden Kapiteln sind das Transaktionskonzept und die Funktionsweise der Currency-Tabelle beschrieben.

Für das Programmieren Ihrer Anwendungen finden Sie die Funktionen der einzelnen DML-Anweisungen am Beispiel COBOL-DML erklärt. Die Anwendung und das Testen der DML finden Sie in den darauf folgenden Kapiteln. Die Nachschlageteile sind getrennt nach

– COBOL-DML, die in den Sprachumfang der ANSCOBOL-Compiler COBOL85 und COBOL2000 integriert ist, und

– CALL-DML, die über die CALL-Schnittstelle der verschiedenen Programmiersprachen angesprochen wird.

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
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

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

Thema	Handbuch	Kapitel
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 REORG-PPP 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.

1.5.3 SDF-Syntaxdarstellung

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

Tabelle 4: Metasyntax

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

Tabelle 5: Datentypen

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

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

Tabelle 6: Zusätze zu Datentypen

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

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

Kennzeichnung	Bedeutung	Beispiele
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Schlüsselwörter. Einige Schlüsselwörter beginnen mit *	OPEN DATABASE COPY-NAME = *NONE
=	Das Gleichheitszeichen verbindet einen Operandennamen mit den dazugehörenden Operandenwerten.	CONFIGURATION-NAME = <name 1..8>
< >	Spitze Klammern kennzeichnen Variablen, deren Wertevorrat durch Datentypen und ihre Zusätze beschrieben wird (Tabellen 24 und 27).	DATABASE = <dbname>

Tabelle 4: Metasyntax (Teil 1 von 2)

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

Tabelle 4: Metasyntax (Teil 2 von 2)

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

Tabelle 5: Datentypen (Teil 1 von 3)

Datentyp	Zeichenvorrat	Besonderheiten
host	A..Z 0..9 $,#,@ Strukturkennzeichen: Bindestrich	1..8 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen A..Z oder $, #, @; wenn weniger als 8 Zeichen angegeben werden, wird intern mit Unterstrich auf 8 Zeichen aufgefüllt
integer	0..9,+,-	+ bzw. - kann nur erstes Zeichen sein.
kset	A..Z 0..9 $,#,@ Strukturkennzeichen: Bindestrich	1..8 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen A..Z oder $, #, @; wenn weniger als 8 Zeichen angegeben werden, wird intern mit Unterstrich auf 8 Zeichen aufgefüllt.
name	A..Z 0..9 $,#,@	1..8 Zeichen darf nicht nur aus 0..9 bestehen oder darf nicht mit einer Ziffer beginnen.
realmname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A..Z
realmref	0..9	1..3 Zeichen
recordname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A..Z Bei Satzarten mit Searchkey wird empfohlen, max. 26 Zeichen lange Namen zu verwenden, da ansonsten der implizit gebildete Setname (SYS_...) entsprechend der Begrenzung der Namenslänge von Sets gekürzt wird.
recordref	0..9	1..3 Zeichen
schemaname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A..Z

Tabelle 5: Datentypen (Teil 2 von 3)

Datentyp	Zeichenvorrat	Besonderheiten
setname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen A..Z
structured-name	A…Z 0…9 $, #, @ Bindestrich	alphanumerische Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A…Z oder $, #, @
subschemaname	A..Z 0..9 Strukturkennzeichen: Bindestrich	1..30 Zeichen Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A..Z
time	0..9 Strukturkennzeichen: Doppelpunkt	Angabe einer Tageszeit Eingabeformat: {hh:mm:ss | hh:mm | hh} hh, mm, ss: bei Stunden, Minuten und Sekunden können führende Nullen weggelassen werden
userid	A..Z 0..9 $,#,@	1..8 Zeichen beginnend mit A..Z oder $,#,@ BPRIVACY: Wenn weniger als 8 Zeichen angegeben werden, wird intern mit Unterstrich auf 8 Zeichen aufgefüllt.
volume	A..Z 0..9 $,#,@	1..6 Zeichen beginnend mit A..Z oder 0..9
x-string	Sedezimal: 00..FF	1..8 Zeichen ist in Hochkommata einzuschließen; der Buchstabe X muss vorangestellt werden; die Anzahl der Zeichen darf ungerade sein.

Tabelle 5: Datentypen (Teil 3 von 3)

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

Tabelle 6: Zusätze zu Datentypen

1.6 Beispieldatenbank

Zu den Beispielen in diesem Handbuch wurde folgende Datenbank verwendet:

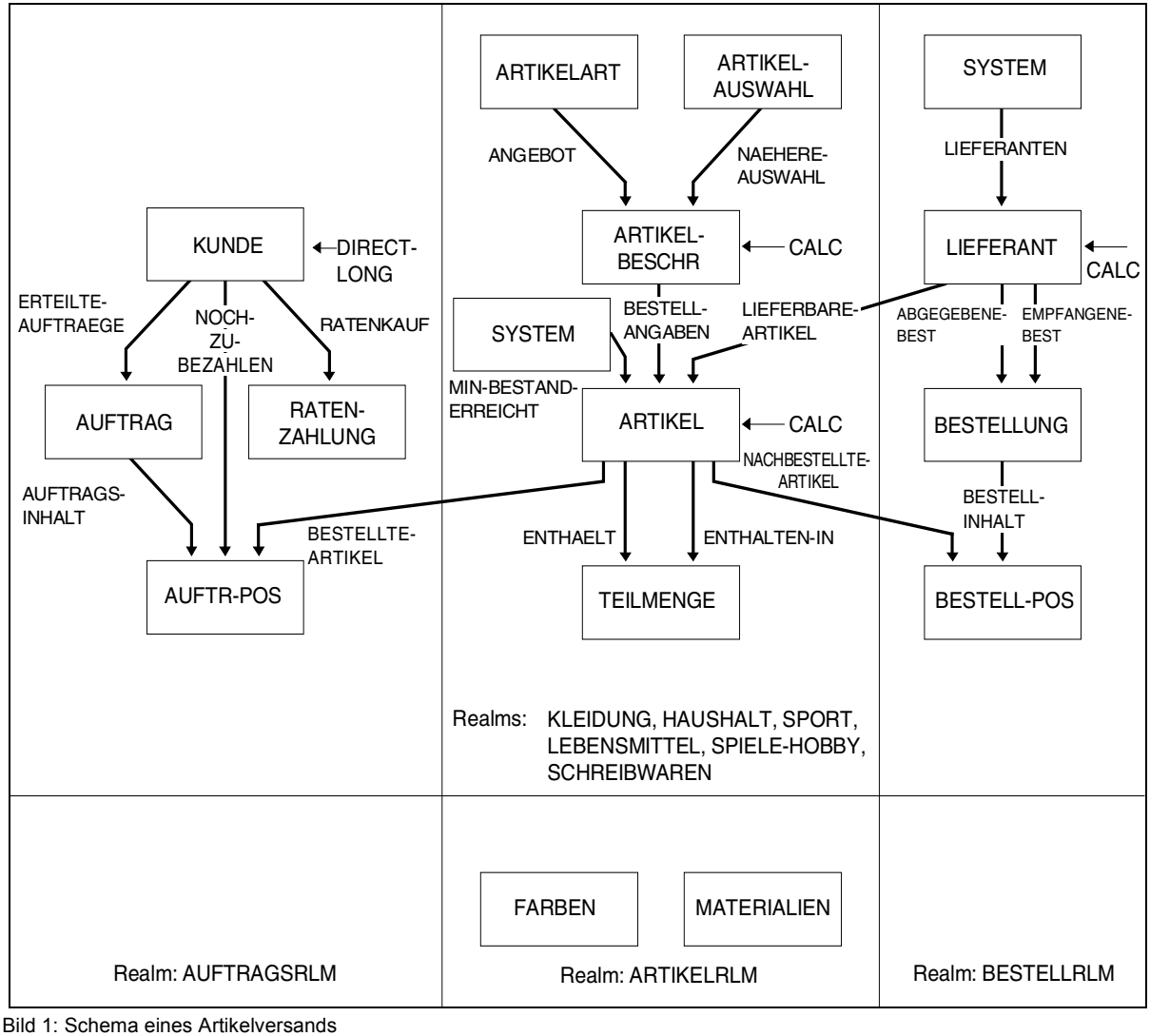


Bild 1: Schema eines Artikelversands

2 Überblick

2.1 Sprachkonzept der DML

Die Sprachmittel, die für die Datenverarbeitung mit Dateien und Dateisystemen vorhanden sind, reichen nicht aus, wenn Sie Datenbankanwendungen programmieren wollen.

Für das Datenbanksystem UDS/SQL wurde die DML (Data Manipulation Language) eingeführt, die dem CODASYL-Sprachkonzept für Datenbanken entspricht. Sie ist nicht auf eine Programmiersprache ausgerichtet, sondern kann in Verbindung mit verschiedenen Programmiersprachen eingesetzt werden.

Die DML existiert in zwei verschiedenen Formen:

– COBOL-DML ist in den Sprachumfang des ANSCOBOL-Compilers COBOL85 und COBOL2000 aufgenommen

– CALL-DML können Sie über die CALL-Schnittstelle der Programmiersprachen ASSEMBLER, COBOL, C, FORTRAN, PASCAL und PL/1 ansprechen

Die Anweisungen der DML berücksichtigen die Beziehungen zwischen den Sätzen, wie sie im Schema einer Datenbank definiert sind. Sie führen die Datenbankänderungen durch, die Sie im Verständigungsbereich Ihres Programms vorbereitet haben. Die sonstige Bearbeitung der Daten müssen Sie in Ihrem Anwenderprogramm mit der gewählten Programmiersprache durchführen. Der Verständigungsbereich in Ihrem Programm ist die UWA (User Work Area).

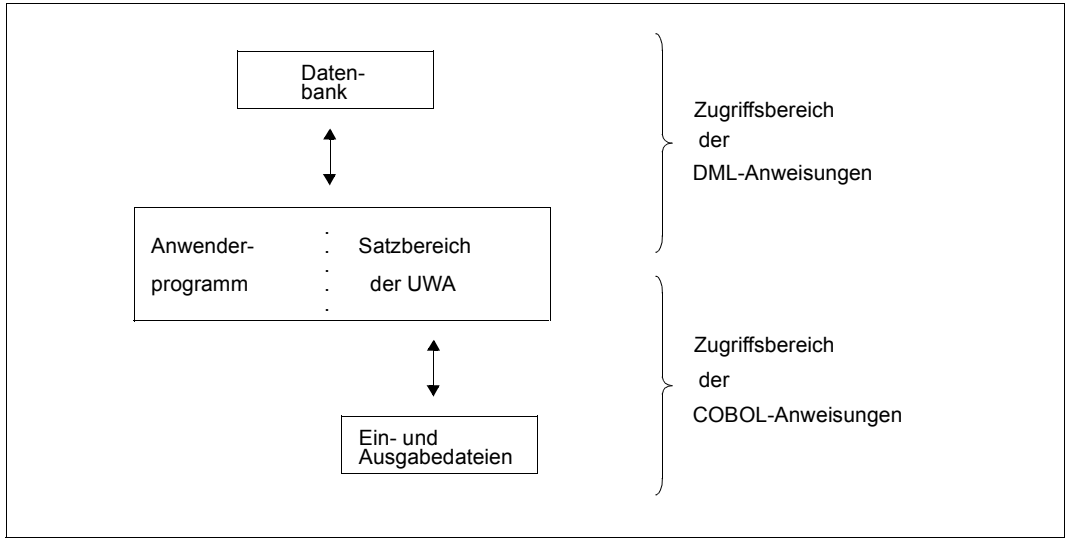


Bild 2: Zugriffsbereich von COBOL- und DML-Anweisungen

2.2 Funktionsumfang der DML

Die DML bietet folgende Funktionen:

- Transaktionen eröffnen und schließen
- Daten wiedergewinnen
- Daten ändern
- Sätze schützen
- Bedingungen abfragen

Diese Funktionen können Sie nur dann ansprechen, wenn sichergestellt ist, dass Sie sich beim Ändern in der Datenbank nicht mit anderen Anwendern gegenseitig stören.
In UDS/SQL sorgt dafür das Transaktionskonzept und das damit verbundene Sicherungskonzept.

3 Transaktionskonzept

Eine Transaktion besteht aus einer logisch abgeschlossenen Folge von DML-Anweisungen. Sie beginnt mit READY und endet mit FINISH. In dieser Folge von Anweisungen behandeln Sie zusammengehörige Aufgaben. Eine Transaktion soll so kurz wie möglich sein, um einzelne Realms und Seiten der Datenbank(en) nicht zu lange zu blockieren. Eine Transaktion sollte möglichst im Batch die Länge von 2000 DML-Anweisungen nicht überschreiten. In einer Online-Anwendung sollten es nicht mehr als 20 DML-Anweisungen sein. Mit dem Transaktionskonzept von UDS/SQL wird sichergestellt, dass verschiedene Anwender mit der (den) Datenbank(en) arbeiten können, ohne sich gegenseitig zu behindern.

Das Transaktionskonzept sorgt für die logische Konsistenz der Datenbank und die Sicherheit der Daten.

Die Transaktion wird entweder vollständig ausgeführt oder bei einem Fehler innerhalb einer Transaktion überhaupt nicht. Dadurch befinden sich angesprochene Datenbanken immer in einem konsistenten Zustand.
Für jede Transaktion werden auch Schutzmaßnahmen durchgeführt, die Sie beim Start der Transaktion je nach Art Ihrer Anwendung selbst bestimmen.

Beim Eröffnen der Transaktion legen Sie als Anwender fest, mit welchen Realms Sie arbeiten und in welcher Art (USAGE-MODE) Sie auf die Datenbank zugreifen wollen. Sie bestimmen, ob noch andere Anwender mit den Realms arbeiten dürfen und schützen damit Ihre Daten vor unberechtigten Zugriffen.

Wenn Sie mit dem linked-in DBH arbeiten, sind die Schutzfunktionen unwichtig, da Sie der einzige Anwender sind.

3.1 Transaktion im Multi-DB-Betrieb

Im Multi-DB-Betrieb bilden mehrere Datenbanken die Multi-DB-Konfiguration. Eine Transaktion kann alle Datenbanken dieser Konfiguration ansprechen.

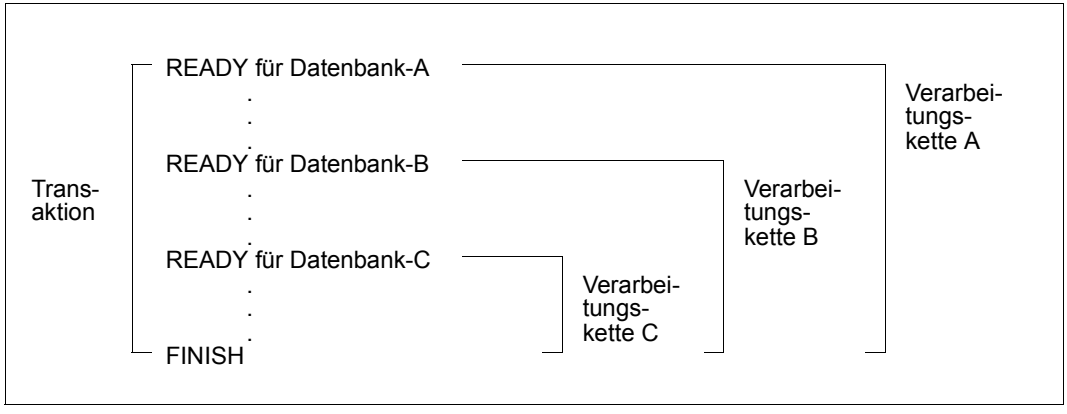


Bild 3: Transaktion und Verarbeitungsketten im Multi-DB-Betrieb

Sie müssen jede Datenbank mit einer READY-Anweisung eröffnen. Deshalb gibt es im Multi-DB-Betrieb innerhalb einer Transaktion mehrere READY-Anweisungen. Die erste READY-Anweisung eröffnet die Transaktion und gleichzeitig die erste Verarbeitungskette. Jede weitere READY-Anweisung eröffnet eine weitere Verarbeitungskette. Eine Verarbeitungskette ist eine Folge von DML-Anweisungen an eine einzelne Datenbank innerhalb einer Transaktion. Alle Verarbeitungsketten werden durch eine gemeinsame FINISH-Anweisung geschlossen. Gleichzeitig wird die Transaktion beendet.

Eine Transaktion besteht im Multi-DB-Betrieb aus mehreren Verarbeitungsketten, die bei COBOL-DML in jeweils einem Anwendermodul verarbeitet werden.

Sie können in einer Transaktion Verarbeitungsketten mit COBOL-DML- und CALL-DML-Anweisungen verwenden. Innerhalb einer Verarbeitungskette dürfen aber nur entweder COBOL-DML- oder CALL-DML-Anweisungen vorkommen.

Ist eine READY-Anweisung innerhalb der Transaktion fehlerhaft, wird die gesamte Transaktion zurückgesetzt (Rollback).

3.2 Transaktion im Mono-DB-Betrieb

In diesem Anwendungsfall entspricht die einzige Verarbeitungskette der Transaktion (siehe Bild 3). Es gibt nur eine READY-Anweisung pro Transaktion.

3.3 Rollback

Rollback setzt die Änderungen der Transaktion zurück, wenn z.B. vor Ende des Programms die Transaktion nicht mit FINISH abgeschlossen werden konnte.

In einigen Situationen muss von UDS/SQL statt einer aufgerufenen DML-Anweisung ein Rollback der gesamten Transaktion durchgeführt werden.
Dies ist z.B. der Fall, wenn sich zwei oder mehrere Transaktionen gegenseitig dadurch blockieren, dass sie auf gesperrte Seiten, Realms oder andere Betriebsmittel (Tabellen, Tasks) zugreifen wollen. Man spricht hierbei von „Deadlock“. Eine der Transaktionen, die den Deadlock verursacht haben, wird zurückgesetzt.

Zum Rollback einer Transaktion führen außerdem:

– schwerwiegende Anwenderfehler

– vorzeitige Beendigung durch den DB-Administrator (DAL-Kommando ABORT, CLOSE CALLS)

– Ein-/Ausgabefehler auf Realms oder Log-Dateien

Einen Rollback müssen Sie im Anwenderprogramm erkennen (siehe Abschnitt „Fehlerbehandlung“ auf Seite 114). Zusätzlich können Sie in den DECLARATIVES eines COBOL-DML-Programms eine USE-Anweisung mit den entsprechenden Datenbanksonderzuständen angeben und eine Behandlungsroutine definieren (siehe Abschnitt „Datenbanksonderzustände“ auf Seite 117).

3.4 Seitenzugriffsschutz

Ein wesentlicher Gesichtspunkt bei der Einführung von Datenbanksystemen ist die Zentralisation von Daten in einer Datenbank. Dies hat andererseits zur Folge, dass parallele Transaktionen gleichzeitig auf die Datenbank zugreifen.

Während bei Lesezugriffen die Datenbank von mehreren Anwendern gleichzeitig benutzt werden kann, ist beim Ändern immer abwechselnder Zugriff auf die zu ändernden Daten notwendig.

Der DBH muss deshalb Vorsorgemaßnahmen treffen, um den korrekten Zustand der Daten sicherzustellen und einen fehlerfreien Ablauf zu garantieren.

Beispiel

Zwei Programme addieren jeweils den Wert 50 zu einem Feld in der Datenbank, das den Anfangswert 100 hat. Das Ergebnis ist nur dann 200, wenn das zweite Programm seinen Zugriff erst dann ausführen kann, nachdem das erste Programm den Wert 150 bereits zurückgeschrieben hat. Wenn beide Programme gleichzeitig lesen und ändern könnten, wäre das Ergebnis möglicherweise 150.

Als Hilfsmittel zur Erhaltung der Konsistenz gibt es zwei Sperr-Ebenen:

– Sperren auf Realm-Ebene (siehe Abschnitt „Transaktion oder Verarbeitungskette eröffnen (READY)“ auf Seite 43)

– Sperren auf Seitenebene (FIND, FETCH, Ändern von Daten, KEEP, FREE);
eine Seite wird gesperrt, sobald auf einen Satz dieser Seite zugegriffen wird.

Auf jeder Sperr-Ebene gibt es zwei Arten von Sperren:

– exklusives Sperren:
Nur eine einzige Transaktion hat Zugriff auf die Daten.

– teilhabendes Sperren:
Mehrere Transaktionen können parallel auf die Daten zugreifen, sofern die Daten nicht exklusiv gesperrt sind.

Mit Hilfe der genannten Sperren auf Seitenebene arbeitet der Seitenzugriffsschutz wie folgt:

– Bei Datenwiedergewinnung (FIND/FETCH) sperrt der Seitenzugriffsschutz eine noch nicht gesperrte Seite gegen den ändernden Zugriff anderer Transaktionen (teilhabendes Sperren).

– Bei Datenänderung sperrt der Seitenzugriffsschutz eine Seite, in der ein Satz geändert wird, exklusiv gegen den Zugriff anderer Transaktionen (exklusives Sperren). Wenn dabei Seiten mit Sekundärdaten (z.B. Tabelleneinträgen) geändert werden, sperrt der Seitenzugriffsschutz diese Seiten ebenfalls.

Auf diese Weise werden Deadlocks bei nur lesenden Anwendungen verhindert.

Der Seitenzugriffsschutz wirkt maximal bis zum Ende der Transaktion.

In Fällen, in denen auf Realm-Ebene kein konkurrierender Zugriff (bei READY EXCLUSIVE) oder keine Datenbankänderung (bei READY PROTECTED RETRIEVAL) möglich ist, werden die Routinen für den Seitenzugriffsschutz ausgeschaltet; bei READY PROTECTED UPDATE entfällt teilhabendes Sperren nur für die Änderungstransaktionen. Das Sperren findet dann in Übereinstimmung mit dem READY-Modus (siehe READY-Anweisung auf Seite 43) nur noch auf Realm-Ebene statt.

Um zu verhindern, dass sich gleichzeitig laufende Transaktionen unnötig stark durch langfristige Sperren gegenseitig behindern, sollte bei Dialogbetrieb, soweit möglich und sinnvoll, bereits nach wenigen Änderungsoperationen die Transaktion mit FINISH beendet und anschließend mit READY eine neue Transaktion eröffnet werden.

Die Seiten der Free Place Administration Table (FPA) und der Database Key Translation Table (DBTT) unterliegen nicht dem Verfahren für den Seitenzugriffsschutz. Die Einträge dieser Tabellen beziehen sich immer auf eine ganz bestimmte Datenseite und können erst dann geändert werden, wenn diese Datenseite geändert und damit gesperrt wurde. Ein explizites Sperren dieser Tabelleneinträge wird dadurch überflüssig. Mehrere Transaktionen können zur selben Zeit unterschiedliche Einträge in einer FPA- oder DBTT-Seite ändern.

4 Currency-Tabelle

In der Currency-Tabelle trägt der DBH gewisse aktuelle Database-Key-Werte ein. Mit den Informationen der Currency-Tabelle können Sie z.B. auf den Satz einer Satzart zugreifen, dessen Database-Key-Wert hier gespeichert ist.
Die Currency-Tabelle wird immer pro Verarbeitungskette geführt und bei FINISH zurückgesetzt.

Pro Satzart, Set und Realm eines Subschemas und für die Verarbeitungskette ist in der Currency-Tabelle je ein Database-Key-Wert enthalten. Die zu diesen Database-Key-Werten gehörenden Sätze heißen:

– CRR (Current Record of Record): aktueller Satz der Satzart

– CRS (Current Record of Set): aktueller Satz des Set

– CRA (Current Record of Area): aktueller Satz des Realm

– CRU (Current Record of Rununit): aktueller Satz der Verarbeitungskette

Ein Satz wird zum aktuellen Satz in der Currency-Tabelle, indem er bei den DML-Anweisungen FIND, STORE, ERASE, CONNECT und MODIFY angesprochen wird.

Die Einträge der Currency-Tabelle werden auch zur Ausführung der FIND NEXT-Anweisung verwendet. Ein FIND NEXT (FIND-4) liefert jeweils den logisch auf den Current Record folgenden Satz einer Satzart, einer Set-Occurrence oder innerhalb eines Realm.

Eine ausführliche Beschreibung zu Aufbau und Wertebereich von Database-Key-Werten finden sie im Handbuch „Entwerfen und Definieren“. Wenn im beschreibenden Text des vorliegenden Handbuchs Einschränkungen hinsichtlich des Aufbaus (DATABASE-KEY/ DATABASE-KEY-LONG) oder des Wertebereichs von Database-Key-Werten bestehen, wird an den entsprechenden Stellen gesondert darauf hingewiesen.

Beispiele

1. Mit einer FIND-4-Anweisung können Sie den Satz auswählen, der auf den CRS folgt.
2. Mit einer FIND-5-Anweisung können Sie die Informationen der Currency-Tabelle auf einen früheren Stand zurücksetzen.
3. Bei der FIND-7-Anweisung

   ```
   FIND PERSONAL-SATZ WITHIN PERSONAL-SET CURRENT USING TAETIGKEIT
   ```

   bestimmt der DBH die Set-Occurrence, aus der der PERSONAL-SATZ auszuwählen ist, über die Currency-Information. Er prüft, welcher Satz der Current Record im Set PERSONAL-SET ist. Aus der Set-Occurrence, zu der der Current Record gehört, wird der PERSONAL-SATZ ausgewählt.

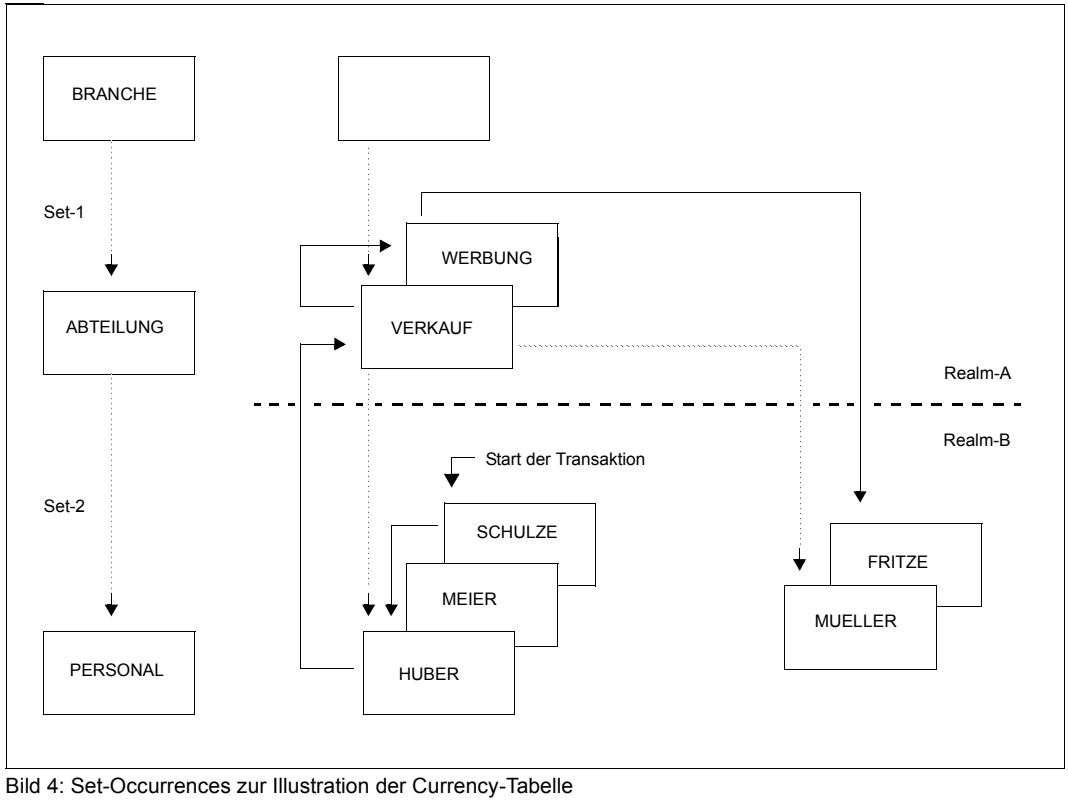


Bild 4: Set-Occurrences zur Illustration der Currency-Tabelle

	Database Key des CRU	Database Key des CRA		Database Key des CRS		Database Key des CRR		
		REALM-A	REALM-B	SET-1	SET-2	BRANCHE	ABTEILUNG	PERSONAL
READY	0	0	0	0	0	0	0	0
Wieder-gewinnen von								
SCHULZE	SCHULZE	0	SCHULZE	0	SCHULZE	0	0	SCHULZE
HUBER	HUBER	0	HUBER	0	HUBER	0	0	HUBER
VERKAUF	VERKAUF	VERKAUF	HUBER	VERKAUF	VERKAUF	0	VERKAUF	HUBER
WERBUNG	WERBUNG	WERBUNG	HUBER	WERBUNG	WERBUNG	0	WERBUNG	HUBER
FRITZE	FRITZE	WERBUNG	FRITZE	WERBUNG	FRITZE	0	WERBUNG	FRITZE
FINISH	0	0	0	0	0	0	0	0

Tabelle 7: Änderungen in der Currency-Tabelle

Bei einigen DML-Anweisungen können Sie angeben, ob eine automatische Änderung der Informationen der Currency-Tabelle unterdrückt werden soll. Das geschieht mit der RETAINING-Angabe (siehe z.B. Abschnitt „Daten wiedergewinnen“ auf Seite 46).

Sie können die Änderung aller Informationen der Currency-Tabelle außer der des CRU unterdrücken.

5 Funktionen der DML

Die Funktionen der DML gliedern sich in fünf Gruppen:

- Transaktionen eröffnen und schließen
- Daten wiedergewinnen
- Daten ändern
- Sätze schützen
- Set-Mitgliedschaften prüfen

Dieses Kapitel behandelt die Umsetzung von Aufgaben in DML-Anweisungen. Die Funktionen werden anhand der Anweisungen der COBOL-DML erklärt.

Die folgende Tabelle zeigt die Zuordnung der entsprechenden CALL-DML-Formate zu den COBOL-DML-Anweisungen:

COBOL-DML	**CALL-DML**
ACCEPT	ACCPTC, ACCPTL
CONNECT	CONNEC
DISCONNECT	DISCON
ERASE	ERASEC
FETCH1-7	FTCH1, FTCH1L, FTCH2-7
FIND1-7	FIND1, FIND1L, FIND2-7
FINISH	FINISC
FREE	FREEC
GET	GETC
IF	IFC
KEEP	KEEPC
-	LOOKC
MODIFY	MODIF1
-	MODIF2
READY	READYC
SET	-
STORE	STORE1, STOR1L
-	STORE2, STOR2L
USE	-

Tabelle 8: Gegenüberstellung von COBOL-DML-Anweisungen und CALL-DML-Formaten

Die Anweisungen SET und USE sind im Abschnitt „Besonderheiten der COBOL-DML“ auf Seite 86 beschrieben. Die LOOKC-Funktion ist im Abschnitt „Besonderheiten der CALL-DML“ auf Seite 92 und ausführlich im Abschnitt „LOOKC-Funktion“ auf Seite 266 beschrieben. Die Syntaxregeln und Formate der COBOL-DML sind im Kapitel „Nachschlageteil der COBOL-DML“ auf Seite 129 beschrieben. Die Syntaxregeln und Parameterdefinitionen der CALL-DML sind im Kapitel „Nachschlageteil der CALL-DML“ auf Seite 197 beschrieben.

5.1 Transaktion eröffnen und schließen

Die Verarbeitung können Sie wie folgt steuern:

– eine Transaktion mit ihren Benutzungsarten eröffnen (READY)

– die Transaktion schließen (FINISH)

5.1.1 Transaktion oder Verarbeitungskette eröffnen (READY)

```
READY[ realmname,...][ USAGE-MODE IS[ {EXCLUSIVE  } ] {RETRIEVAL} ]
                                      {PROTECTED  }   {UPDATE   }
```

Die READY-Anweisung eröffnet eine Transaktion oder bei Multi-DB eine Verarbeitungskette für eine Datenbank. READY legt die Benutzungsarten fest und bestimmt, mit welchen Realms gearbeitet wird.

Pro Datenbank, auf die Sie zugreifen wollen, müssen Sie eine READY-Anweisung angeben. Der erste READY eröffnet gleichzeitig die Transaktion.

Die Realm-Namen, die Sie angeben, müssen sich auf die Realms in Ihrem Subschema beziehen.

Mit USAGE-MODE legen Sie die Benutzungsart Ihrer Realms fest:

- für den Zugriff anderer Verarbeitungsketten, die ebenfalls mit den Realms arbeiten wollen:

  EXCLUSIVE: Keine weitere Verarbeitungskette darf gleichzeitig mit den Realms arbeiten.

  PROTECTED: Andere Verarbeitungsketten dürfen nicht gleichzeitig ändern.

- für den Zugriff Ihrer Verarbeitungskette:

  RETRIEVAL: Daten wiedergewinnen.

  UPDATE: Daten wiedergewinnen und ändern.

Geben Sie USAGE-MODE nicht an, wird USAGE-MODE RETRIEVAL angenommen.

Aus der Kombination von beiden Angaben ergeben sich für eine Verarbeitungskette sechs mögliche Benutzungsarten, von denen sich nicht jede mit den Benutzungsarten anderer Verarbeitungsketten verträgt.

Welche Benutzungsarten auf einem Realm bei Mono-DB-Betrieb gleichzeitig zugelassen sind, können Sie der folgenden Tabelle entnehmen:

	RETRIEVAL	UPDATE	PROTECTED RETRIEVAL	PROTECTED UPDATE	EXCLUSIVE RETRIEVAL	EXCLUSIVE UPDATE
RETRIEVAL	X[1]	(X)[2]	X	(X)	–[3]	–
UPDATE	(X)	(X)	–	–	–	–
PROTECTED RETRIEVAL	X	–	X	–	–	–
PROTECTED UPDATE	(X)	–	–	–	–	–
EXCLUSIVE RETRIEVAL	–	–	–	–	–	–
EXCLUSIVE UPDATE	–	–	–	–	–	–

Tabelle 9: Erlaubte Kombinationen von USAGE-MODE bei Mono-DB-Betrieb

1 X möglich

2 (X) Deadlock möglich

3 - nicht möglich

Beispiel

Wenn ein Realm schon mit USAGE-MODE UPDATE oder EXCLUSIVE RETRIEVAL eröffnet ist, kann ihn eine weitere Verarbeitungskette nicht mit READY USAGE-MODE PROTECTED UPDATE eröffnen.

Die READY-Anweisung wird nur dann ausgeführt, wenn die vereinbarte Benutzungsart zusammenpasst mit der Benutzungsart anderer Verarbeitungsketten, die die gleichen Realms ansprechen. Andernfalls wartet das System mit der Ausführung, bis die nicht passenden Verarbeitungsketten beendet sind.

Innerhalb einer Transaktion dürfen nicht zwei Verarbeitungsketten parallel auf eine einzelne Datenbank zugreifen.
Diese Einschränkung verhindert, dass sich zwei Verarbeitungsketten des gleichen Benutzers durch ihre Zugriffsrechte auf Realm-Ebene gegenseitig sperren (Deadlock).

5.1.2 Transaktion schließen (FINISH)

```
FINISH[ WITH CANCEL]
```

Eine Transaktion beenden Sie mit FINISH.
Gleichzeitig beenden Sie damit alle offenen Verarbeitungsketten dieser Transaktion und schließen alle Realms, die von diesen Verarbeitungsketten eröffnet wurden. Bei der Beendigung der Transaktion werden die Before-Images entwertet und die Currency-Tabellen gelöscht. Der Temporäre Realm wird freigegeben. Ein erweiterter Satzschutz durch KEEP wird aufgehoben.

Beenden Sie die Transaktion z.B. bei einem Testlauf des Programms mit WITH CANCEL, macht der DBH mit Rollback alle Änderungen der Verarbeitungsketten dieser Transaktion rückgängig. Falls keine RLOG-Dateien vorhanden sind, z.B. bei PP LOG=NO, oder bei defekten RLOG-Dateien, wird die betroffene Datenbank als defekt gekennzeichnet und solange für alle Transaktionen gesperrt, bis der Datenbankadministrator die Datenbank in einen konsistenten Zustand gebracht hat, wobei auch vorhergehende Transaktionen wieder rückgängig gemacht werden.

5.2 Daten wiedergewinnen

Für das Wiedergewinnen von Daten aus der Datenbank stehen Ihnen vier DML-Anweisungen zur Verfügung:

FIND sucht einen Satz in der Datenbank auf, trägt seinen Database-Key-Wert in die Currency-Tabelle ein und kennzeichnet ihn damit als aktuellen Satz. Mit der Anweisungsfolge ACCEPT, FIND-1 kann UDS/SQL auf diesen Satz solange direkt über den Database-Key-Wert zugreifen, wie der Satz in der Currency-Tabelle verzeichnet ist.

GET liefert den Satz, der in der Currency-Tabelle als aktueller Satz (CRU) vermerkt ist, an das Anwenderprogramm in die UWA.

FETCH
kombiniert die Funktion von FIND und GET, sucht also einen Satz nicht nur in der Datenbank und trägt seinen Database-Key-Wert in die Currency-Tabelle ein, sondern liefert den Satz auch in die UWA des Anwenderprogramms.

ACCEPT
liefert zu einem in der Currency-Tabelle vermerkten Satz den Database-Key-Wert oder den Namen des Realm, in dem der Satz gespeichert ist, an das Anwenderprogramm.

Diese DML-Anweisungen sind in den folgenden Abschnitten beschrieben. Dabei wird nicht mehr zwischen FIND und FETCH unterschieden. Dass FETCH den zuletzt angesprochenen Satz zusätzlich an das Anwenderprogramm liefert, wird immer als bekannt vorausgesetzt.

Satz in der Datenbank auffinden

```
{FIND  } satzauswahlausdruck[ RETAINING CURRENCY .....]
{FETCH }
```

Mit *satzauswahlausdruck* definieren Sie, wie Sie auf Sätze zugreifen wollen.
Die Erläuterung der Zugriffsmöglichkeiten ist Gegenstand der anschließenden Kapitel.

UDS/SQL kennzeichnet einen ausgewählten Satz, indem es alle betroffenen Spalten der Currency-Tabelle mit dem Database-Key-Wert des Satzes überschreibt. Der Database-Key-Wert eines früher ausgewählten Satzes verschwindet somit aus diesen Spalten der Currency-Tabelle: Dieser Satz kann dann nur mit einem neuen Suchvorgang in der Datenbank wiedergewonnen werden.

Mit RETAINING können Sie das Aktualisieren der Currency-Tabelle gezielt dort unterdrücken, wo Sie den Database-Key-Wert eines vorhergehenden Satzes bewahren wollen.

Aktualisieren der Currency-Tabelle gezielt steuern

```
                                  ┌ MULTIPLE                                  ┐
RETAINING CURRENCY FOR ┤                                                     ├
                                  └ [ REALM][ RECORD][ ┌SETS          ┐][ ] ┘
                                                       └setname,...   ┘
```

Mit diesem Zusatz zu einer FIND/FETCH-Anweisung können Sie das Aktualisieren der Currency-Tabelle außer für den CRU gezielt unterdrücken.

Wenn Sie mit RETAINING das Aktualisieren der Currency-Tabelle unterdrückt haben, können Sie die Currency-Tabelle nachträglich auf den neuesten Stand bringen, ohne den Satz erneut in der Datenbank suchen zu müssen:

```
┌FIND  ┐ CURRENT[ RETAINING CURRENCY FOR[ REALM][ RECORD][ ┌SETS          ┐]]
└FETCH ┘                                                   └setname,...   ┘
```

Diese Anweisung überträgt den Database-Key-Wert des CRU, der ja immer aktuell ist, auf die übrigen betroffenen Spalten der Currency-Tabelle.
Mit RETAINING können Sie auch hier angeben, welche Spalten davon ausgenommen sein sollen.

5.2.1 Direkter Zugriff auf Satzartebene

Mit dieser Zugriffsart wählen Sie einen Satz über den Inhalt eines Feldes bzw. einer Kombination von Feldern aus. Die Auswahlmenge sind alle Sätze einer Satzart. Die Felder können beliebige Felder der Satzart sein.

5.2.1.1 Direkter Zugriff über den Database Key (FIND/FETCH-1)

```
{FIND  }[ satzname] DATABASE-KEY IS feldname [OR {PRIOR}]
{FETCH }                                        {NEXT }
```

Jeder Satz besitzt einen in der gesamten Datenbank eindeutigen Schlüssel, den so genannten Database-Key-Wert.
Wenn Sie ins Feld *feldname* einen Database-Key-Wert übertragen, liefert UDS/SQL den zugehörigen Satz. UDS/SQL kann zusätzlich prüfen, ob der Satz zur Satzart *satzname* gehört.

Das Feld *feldname* muss mit USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG definiert sein.
Wenn Sie nach einem Satz mit einem Database-Key-Wert suchen wollen, der eine REC-REF > 254 und/oder eine RSQ > 2^{24}-1 enthält, muss *feldname* mit USAGE IS DATABASE-KEY-LONG definiert sein.

Wenn Sie OR PRIOR/NEXT angeben, wird der Satz mit dem nächstkleineren/nächstgrößeren Database-Key zur Verfügung gestellt, falls es keinen Satz mit dem angegebenen Database-Key gibt. Ein spezieller Statuscode zeigt dann an, dass nicht der Satz mit dem zunächst angegebenen DATABASE-KEY ausgewählt wurde.

5.2.1.2 Direkter Zugriff über den CALC-Key (FIND/FETCH-2)

```
{FIND  } ANY satzname
{FETCH }
```

Diese FIND-Anweisung dürfen Sie nur benutzen, wenn im Schema ein CALC-Key für die Satzart *satzname* definiert ist (siehe Handbuch „Entwerfen und Definieren“, LOCATION MODE-Klausel). Sie müssen zuvor die Felder, die im Schema als CALC-Key definiert sind, mit dem CALC-Key-Wert des gewünschten Satzes versorgen. UDS/SQL greift auf den Satz zu, indem es diesen Wert durch ein Hashverfahren in eine relative Seitennummer umrechnet.

Wenn die Satzart *satzname* auf mehrere Realms verteilt ist, müssen Sie zuvor außerdem das AREA-ID-Feld (siehe Handbuch „Entwerfen und Definieren“, WITHIN-Klausel) mit dem Namen des Realms versorgen, in dem sich der gewünschte Satz befindet, wenn die Satzart nicht Membersatzart einer verteilbaren Liste ist.

Wenn im Schema DUPLICATES ARE ALLOWED erklärt ist, kann ein CALC-Key-Wert zu mehreren Sätzen der Satzart gehören. Die obige Anweisung liefert jedoch nur einen Satz. Für jeden weiteren Satz der Satzart mit gleichem CALC-Key-Wert im selben Realm müssen Sie eine Anweisung der folgenden Form schreiben:

```
{FIND  }
{      } DUPLICATE satzname
{FETCH }
```

Diese Anweisung ist nur sinnvoll, wenn der CALC-Key der Satzart *satzname* im Schema mit DUPLICATES ARE ALLOWED definiert ist. In dieser Satzart sucht UDS/SQL dann einen vom CRR verschiedenen Satz, der denselben CALC-Key-Wert besitzt wie der CRR und im Realm des CRR liegt. Für die Suche benutzt UDS/SQL das zugehörige Hashverfahren. Für jedes zu suchende Duplikat müssen Sie die Anweisung wiederholen. Dass Sie damit alle Sätze erreichen, die zu einem CALC-Key-Wert gehören, ist nur garantiert, wenn Sie in jedem Realm, in dem Sätze der Satzart liegen, den ersten Satz mit

```
{FIND  }
{      } ANY satzname
{FETCH }
```

suchen.

5.2.1.3 Direkter Zugriff über beliebige Felder (FIND/FETCH-3/7)

Sie haben drei Möglichkeiten, Sätze mit vorgegebenen Feldinhalten zu suchen:

– Sie geben Feldinhalte aus dem CRR vor und suchen ein Duplikat.

– Sie geben die gewünschten Feldinhalte in der UWA an und suchen einen Satz, der diese Feldinhalte besitzt.

– Sie geben die gewünschten Feldinhalte in der UWA oder in der Anweisung selbst an und verknüpfen diese zu einem Suchausdruck, wobei Sie Klammern, die logischen Operatoren UND, ODER, NICHT und die Vergleichsoperatoren >, < und = verwenden können.
In diesem Fall müssen Sie die Feldinhalte nicht komplett vorgeben: Mit Hilfe einer Maske können Sie die Ausschnitte aus einem Feld ausblenden, für die Sie keinen Wert angeben wollen. UDS/SQL sucht alle Sätze, die dem Suchausdruck genügen und macht sie zu Membersätzen in einem impliziten dynamischen Set. Wenn Sie RESULT IN angeben, werden die Sätze zusätzlich Member in einem expliziten dynamischen Set.
Die dynamischen Sets können Sie mit DML-Anweisungen weiterverarbeiten, z.B. um

Sätze nach mehreren Kriterien zu durchsuchen.
Den impliziten dynamischen Set können Sie aufsteigend oder absteigend sortieren (SORTED BY).

Diese Zugriffsarten sind im Folgenden detailliert erläutert.

Feldinhalte aus dem CRR vorgeben

```
{FIND  } DUPLICATE WITHIN satzname USING satzelementname,...
{FETCH }
```

UDS/SQL sucht einen Satz der Satzart *satzname*, der in den angegebenen Satzelementen mit dem CRR dieser Satzart übereinstimmt.

Feldinhalte in der UWA vorgeben

```
{FIND  } satzname USING satzelementname,...[OR {PRIOR }]
{FETCH }                                      {NEXT  }
```

UDS/SQL sucht einen Satz der Satzart *satzname*, der in den angegebenen Satzelementen mit den in der UWA vorgegebenen Werten übereinstimmt. Diese Satzelemente müssen Sie zuvor in der UWA mit den gewünschten Werten versorgen.

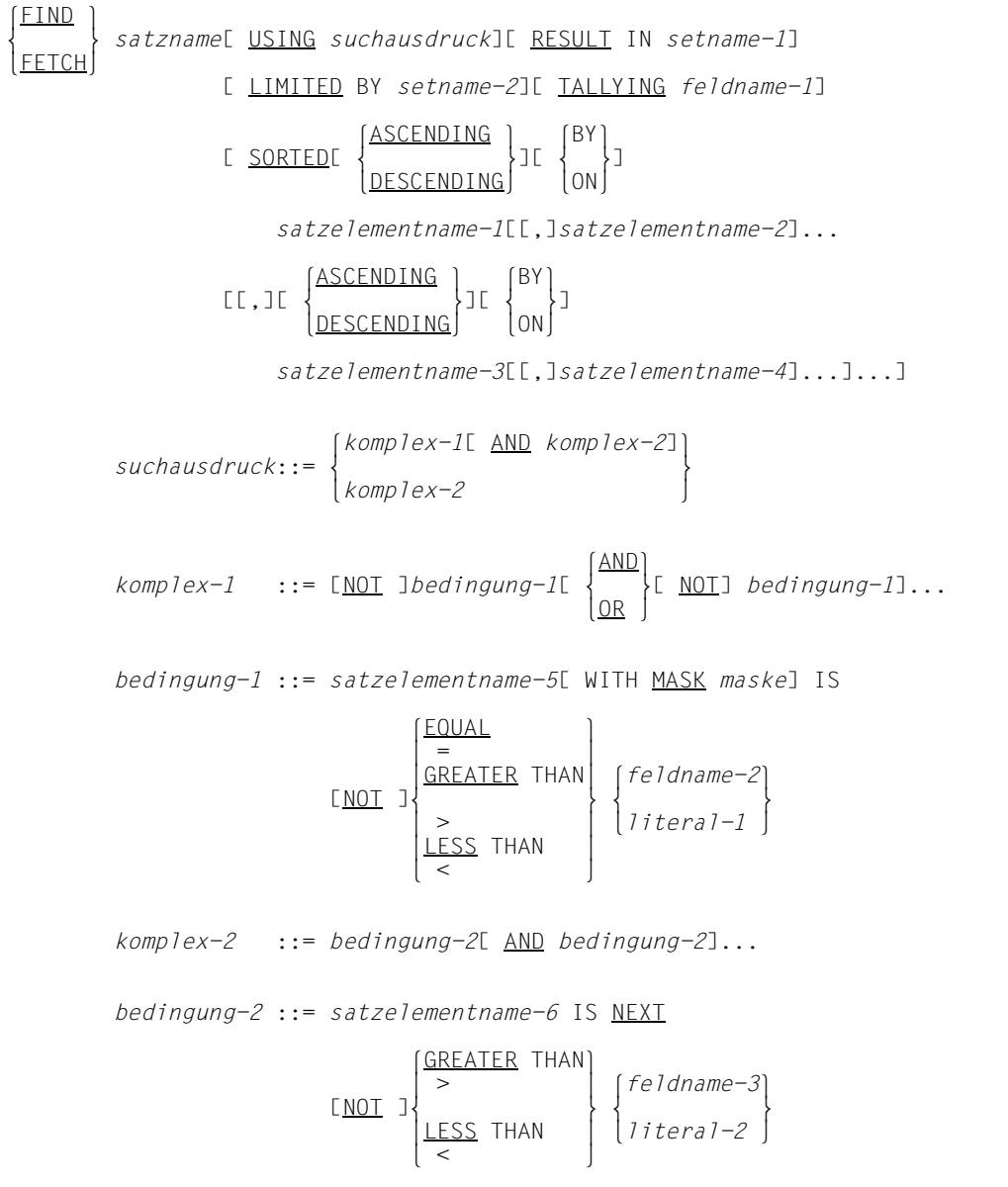
Bei Angabe OR PRIOR/NEXT wird der vorige/nächste Satz als Treffer zur Verfügung gestellt, falls es keinen Satz gibt, der mit den vorgegebenen Werten übereinstimmt. Ein spezieller Statuscode zeigt dann an, dass nicht der Satz mit den vorgegebenen Werten ausgewählt wurde. *satzelementname*,... muss ein SEARCH KEY USING INDEX sein. Der nächste Satz wird durch die Sortierreihenfolge des Schlüssels bestimmt.

Falls es mehrere Sätze gibt, liefert die obige Anweisung jeweils den ersten Satz. Für jeden weiteren Satz müssen Sie die folgende Anweisung benutzen:

```
{FIND  } DUPLICATE WITHIN satzname USING satzelementname,...
{FETCH }
```

Dabei geben Sie dieselben Satzelemente an, die zur Auswahl des ersten Satzes geführt haben. Diese Anweisung ist vorhergehend erklärt (siehe Abschnitt „Feldinhalte aus dem CRR vorgeben“ auf Seite 50).

Feldinhalte zu einem Suchausdruck verknüpfen



Diese Anweisung kann nicht nur einen, sondern alle Sätze der Satzart *satzname* aus der Datenbank auswählen, die dem Suchausdruck genügen.

FIND-Anweisungen, die nur einen Satz aus der Datenbank auswählen, benutzen die Currency-Tabelle, um das Auswahlergebnis zu speichern.
Bei der Auswahl einer Satzmenge speichert UDS/SQL das Auswahlergebnis in einen dynamischen Set, indem es die ausgewählten Sätze zu Membersätzen des dynamischen Set macht. Nur der erste gefundene Satz wird aktueller Satz in der Currency-Tabelle. Der dynamische Set kann Auswahlmenge für weitere FIND-Anweisungen sein.

Im *suchausdruck* geben Sie die Feldinhalte oder Teile von Feldinhalten, die in den zu suchenden Sätzen stehen sollen, durch Vergleichsbedingungen vor. Sie können sowohl mit einem Wert vergleichen, den Sie in der Anweisung selbst angeben, als auch mit einem Feld, das im COBOL-Programm definiert ist.

Bei einem *suchausdruck* der Form *komplex-1* AND *komplex-2* werden zunächst alle Sätze bestimmt, die *komplex-1* erfüllen. Aus der Menge der Sätze, die *komplex-1* erfüllen, werden anschließend die Sätze ermittelt, die die Bedingung(en) in *komplex-2* erfüllen.

Die Bedingungen *bedingung-2* in *komplex-2* werden der Reihe nach ausgewertet, beginnend mit der am weitesten links stehenden Bedingung.

Durch den Zusatz NEXT wird die Treffermenge einer *bedingung-2* in *komplex-2* auf diejenigen Sätze der Satzart *satzname* eingeschränkt, die als *satzelement-6* den Wert W mit den folgenden beiden Eigenschaften enthalten:

– W genügt dem Vergleich in *bedingung-2.*

– Von allen in der Datenbank vorkommenden Werten für *satzelement-6*, die dem Vergleich in *bedingung-2* genügen, liegt W am nächsten bei dem durch *literal-2* bzw. *feldname-3* spezifizierten Vergleichswert.

- Teile von Feldinhalten ausblenden

  Durch WITH MASK *maske* definieren Sie Teile von *satzelementname-5*, deren Inhalt UDS/SQL in einer Vergleichsbedingung nicht berücksichtigen soll.

- Alle Sätze der Satzart in einen dynamischen Set einhängen

  Wenn Sie keinen Suchausdruck angeben, werden alle Sätze der Satzart Membersätze in einem dynamischen Set.

- Auswahlmenge auf einen dynamischen Set begrenzen

  Mit LIMITED BY *setname-2* nennen Sie einen dynamischen Set, auf den UDS/SQL die Suche nach Sätzen beschränken soll. Das heißt, UDS/SQL berücksichtigt dann nur Sätze, die zur Satzart *satzname* gehören und gleichzeitig Membersätze des dynamischen Set sind.

- Die Sätze der ausgewählten Satzmenge zählen

  Mit TALLYING zählen Sie die Sätze, die in der ausgewählten Satzmenge enthalten sind. Ins Feld *feldname-1* überträgt UDS/SQL die Anzahl der gefundenen Sätze.

- Die Sätze der ausgewählten Satzmenge sortieren

  Mit SORTED BY sortieren Sie die Sätze, die in der ausgewählten Satzmenge enthalten sind. *satzelementname-1* bis *satzelementname-4* müssen Elemente der Satzart *satzname* sein. Mit ASCENDING legen Sie eine aufsteigende Reihenfolge fest und mit DESCENDING eine absteigende Reihenfolge.
  Standardwert der ersten Angabe ist ASCENDING, Standardwert der in der Wiederholung stehenden Angabe ist die gerade aktuelle Sortierrichtung. Es gilt die Einschränkung, dass alle Sortierfelder die gleiche Sortierrichtung aufweisen müssen.

- Auf ein Duplikat zugreifen

  Bei der Auswahl einer Satzmenge, speichert UDS/SQL das Auswahlergebnis in einen dynamischen Set. Nur der erste gefundene Satz wird aktueller Satz in der Currency-Tabelle. Ohne Verwendung der RESULT-Klausel müssen Sie für jeden weiteren Satz, den Sie aus dem Auswahlergebnis abrufen wollen, die folgende Anweisung benutzen:

  ```
  { FIND  }
  {       } DUPLICATE WITHIN satzname
  { FETCH }
  ```

  Diese Anweisung bezieht sich immer auf die zuletzt ausgewählte Satzmenge, da der dynamische Set immer nur das Ergebnis des letzten Suchausdrucks enthält.

  satzname muss die Satzart bezeichnen, die mit dem vorhergehenden Suchausdruck bearbeitet wurde. UDS/SQL liefert dann den auf den CRS folgenden Satz des dynamischen Set.

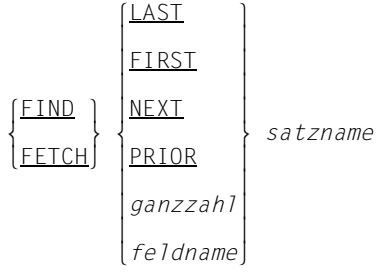
- Das Auswahlergebnis weiterverarbeiten

  UDS/SQL speichert das Auswahlergebnis eines Suchausdrucks in einen impliziten dynamischen Set. Dies ist ein Set, den Sie nicht mit einem Namen ansprechen können.

  Mit RESULT IN *setname-1* nennen Sie einen dynamischen Set, der bereits im Schema definiert ist und das Ergebnis eines Suchausdrucks aufnehmen soll. Das in diesem dynamischen Set gespeicherte Ergebnis können Sie mit FIND/FETCH-4-Anweisungen auf Setebene weiterverarbeiten.

5.2.2 Sequenzieller Zugriff auf Satzartebene (FIND/FETCH-4)

Mit dieser Zugriffsart wählen Sie einen Satz auf Grund der Position aus, die er innerhalb der logischen Reihenfolge aller Sätze einer Satzart einnimmt.
Die Reihenfolge ist durch aufsteigende Database-Key-Werte bestimmt.



Aus der Satzart *satzname* können Sie den letzten oder den ersten Satz, den Nachfolger oder den Vorgänger des CRR oder den Satz auswählen, dessen Position einem anzugebenden Zahlenwert entspricht.

5.2.3 Zugriff auf den CRR (FIND/FETCH-5)

Hiermit greifen Sie auf den CRR der Satzart *satzname* zu und setzen damit die Currency-Information auf einen früheren Stand zurück.
Der CRR wird wieder zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle bzw. dort, wo Sie das Aktualisieren nicht mit RETAINING unterdrücken.

5.2.4 Direkter Zugriff auf Setebene (FIND/FETCH-3/7)

Mit dieser Zugriffsart wählen Sie einen Satz über den Inhalt eines Feldes bzw. einer Kombination von Feldern aus. Die Auswahlmenge sind alle Sätze einer Set-Occurrence. Die Felder können beliebige Felder der Membersatzart sein.
UDS/SQL benutzt immer die zum CRS gehörige Set-Occurrence als Auswahlmenge.

Sie haben drei Möglichkeiten, Sätze mit vorgegebenen Feldinhalten zu suchen:

- Sie geben Feldinhalte aus dem CRS vor und suchen ein Duplikat (FIND-3).
- Sie geben die gewünschten Feldinhalte in der UWA an und suchen einen Satz, der diese Feldinhalte besitzt.
- Sie geben die gewünschten Feldinhalte in der UWA oder in der Anweisung selbst an und verknüpfen diese zu einem Suchausdruck, wobei Sie Klammern, die logischen Operatoren UND, ODER, NICHT und die Vergleichsoperatoren >, < und = verwenden können.
  In diesem Fall müssen Sie die Feldinhalte nicht komplett vorgeben. Mit Hilfe einer Maske können Sie die Ausschnitte aus einem Feld ausblenden, für die Sie keinen Wert angeben wollen. UDS/SQL sucht alle Sätze, die dem Suchausdruck genügen und macht sie zu Membersätzen in einem impliziten dynamischen Set. Wenn Sie RESULT IN angeben, werden die Sätze zusätzlich Member in einem expliziten dynamischen Set.
  Die dynamischen Sets können Sie mit DML-Anweisungen weiterverarbeiten, z.B. um Sätze nach mehreren Kriterien zu durchsuchen.
  Den impliziten dynamischen Set können Sie aufsteigend oder absteigend sortieren (SORTED BY).

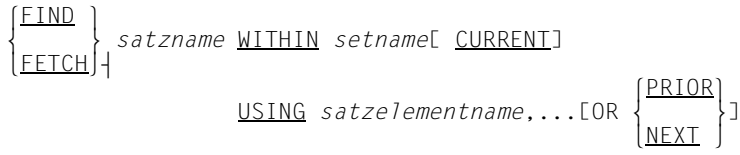
Diese Zugriffsarten sind im Folgenden detailliert erläutert.

Feldinhalte aus dem CRS vorgeben

```
{FIND  }
{      } DUPLICATE WITHIN setname USING satzelementname,...
{FETCH}
```

Aus der Set-Occurrence, die den CRS des Set *setname* enthält, wählt UDS/SQL einen vom CRS verschiedenen Membersatz aus, der in den angegebenen Satzelementen mit dem CRS übereinstimmt.

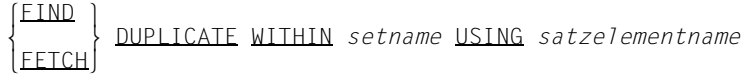
Feldinhalte in der UWA vorgeben



Aus einer Occurrence des Set *setname* wählt UDS/SQL einen Satz aus, der in den angegebenen Satzelementen mit den in der UWA vorgegebenen Werten übereinstimmt. Sie müssen zuvor die Set-Occurrence auswählen und die Satzelemente in der UWA mit den gewünschten Werten versorgen. Die Set-Occurrences werden gemäß der im Schema für diesen Set definierten Auswahlmethode ausgewählt. Ist dort die Auswahlmethode mit SELECTION THRU LOCATION MODE OF OWNER definiert, können Sie diese durch Angabe von CURRENT ausschalten. Die Set-Occurrence können Sie dann durch den CRS auswählen.
setname darf kein dynamischer Set sein.

Bei Angabe OR PRIOR bzw. OR NEXT wird der unmittelbar vorhergehende bzw. der nächste Satz als Treffer zur Verfügung gestellt, falls es keinen Satz gibt, der mit den vorgegebenen Werten übereinstimmt. Ein spezieller Statuscode zeigt dann an, dass nicht der optimale Satz gefunden wurde. *satzelementname*,... muss ein ASCENDING/DESCENDING/SEARCH KEY USING INDEX sein. Der unmittelbar vorhergehende bzw. der nächste Satz wird durch die Sortierreihenfolge des Schlüssels bestimmt.

Falls es mehrere Sätze gibt, die die vorgegebenen Feldinhalte besitzen, und OR PRIOR nicht verwendet wird, können Sie jeden weiteren Satz mit der folgenden Anweisung lesen:



Dabei geben Sie dieselben Satzelemente an, die zur Auswahl des ersten Satzes geführt haben. Diese Anweisung ist vorhergehend erklärt (siehe Abschnitt „Feldinhalte aus dem CRS vorgeben“ auf Seite 55).

Feldinhalte zu einem Suchausdruck verknüpfen

```
{FIND  } satzname WITHIN setname-1[ CURRENT][ USING suchausdruck]
{FETCH }

        [ RESULT IN setname-2][ LIMITED BY setname-3][ TALLYING feldname]

        [ SORTED[ {ASCENDING } ][ {BY} ]
                  {DESCENDING}    {ON}

            satzelementname-1[[,]satzelementname-2]...

        [[,][ {ASCENDING } ][ {BY} ]
              {DESCENDING}    {ON}

            satzelementname-3[[,]satzelementname-4]...]...]

suchausdruck ::= siehe Seite 51
```

Diese Anweisung funktioniert analog dem Direktzugriff mit Hilfe von Suchausdrücken auf Satzartebene (siehe Seite 51).
An dieser Stelle sind deshalb nur die Abweichungen vom Zugriff auf Satzartebene erklärt.

Die Anweisung durchsucht eine Set-Occurrence des Set *setname-1*, die Sie zuvor bestimmen müssen. Die Set-Occurrences werden gemäß der im Schema für diesen Set definierten Auswahlmethode ausgewählt. Ist dort die Auswahlmethode mit SELECTION THRU LOCATION MODE OF OWNER definiert, können Sie diese durch Angabe von CURRENT ausschalten. Die Set-Occurrence können Sie dann durch den CRS auswählen.

- Alle Membersätze der Set-Occurrence in einen dynamischen Set einhängen

  Wenn Sie keinen Suchausdruck angeben, werden alle Membersätze der Set-Occurrence Membersätze in einem dynamischen Set.

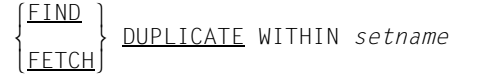
- Zwei Set-Occurrences gleichzeitig durchsuchen

  Mit LIMITED BY *setname-3* nennen Sie einen dynamischen Set, auf den UDS/SQL die Suche nach Sätzen beschränken soll. Das heißt, UDS/SQL berücksichtigt dann nur Sätze, die

  – Membersätze in der ausgewählten Set-Occurrence des Set *setname-1* sind und
  – gleichzeitig Membersätze im angegebenen dynamischen Set sind.

- Auf ein Duplikat zugreifen

  Bei der Auswahl einer Satzmenge speichert UDS/SQL das Auswahlergebnis in einen dynamischen Set. Nur der erste gefundene Satz wird aktueller Satz in der Currency-Tabelle. Ohne Verwendung der RESULT-Klausel müssen Sie für jeden weiteren Satz, den Sie aus dem Auswahlergebnis abrufen wollen, die folgende Anweisung benutzen:

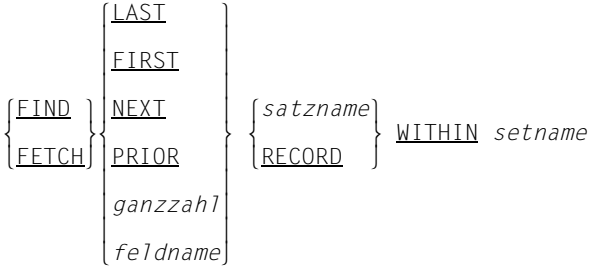
  

  Diese Anweisung bezieht sich immer auf die zuletzt ausgewählte Satzmenge, da der implizite dynamische Set immer nur das Ergebnis des letzten Suchausdrucks enthält.

  setname muss den Set bezeichnen, der mit dem vorhergehenden Suchausdruck bearbeitet wurde. UDS/SQL liefert dann den auf den CRS folgenden Satz des dynamischen Set.

5.2.5 Sequenzieller Zugriff auf Setebene (FIND/FETCH-4)

Mit dieser Zugriffsart wählen Sie einen Satz auf Grund der Position aus, die er innerhalb der logischen Reihenfolge aller Sätze einer Set-Occurrence einnimmt. Die Reihenfolge ist im Schema durch die ORDER-Klausel für diesen Set bestimmt. Der Ownersatz ist in dieser Reihenfolge Vorgänger des ersten und gleichzeitig Nachfolger des letzten Membersatzes.



Aus der Set-Occurrence, die den CRS des Set *setname* enthält, können Sie den letzten oder den ersten Satz, den Nachfolger oder den Vorgänger des CRS oder den Satz auswählen, dessen Position einem anzugebenden Zahlenwert entspricht. Wenn Sie *satzname* angeben, wählt UDS/SQL den Satz nur dann aus, wenn *satzname* die Membersatzart des Set bezeichnet.

5.2.6 Zugriff auf den CRS (FIND/FETCH-5)

```
{FIND  }
{      } CURRENT[ satzname] WITHIN setname
{FETCH }
```

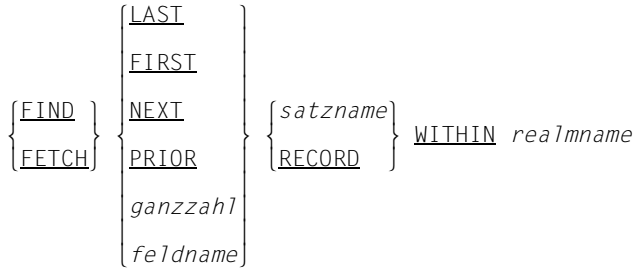
Hiermit greifen Sie auf den CRS des Set *setname* zu und setzen damit die Currency-Information auf einen früheren Stand zurück. Der CRS des Set *setname* wird wieder zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle bzw. dort, wo Sie das Aktualisieren nicht mit RETAINING unterdrücken. Wenn Sie *satzname* angeben, greift UDS/SQL nur dann auf den CRS zu, wenn *satzname* die Membersatzart des Set bezeichnet und der CRS ein Membersatz ist.

5.2.7 Zugriff auf den Owner eines CRS (FIND/FETCH-6)

```
{FIND  }
{      } OWNER WITHIN setname
{FETCH }
```

Diese Anweisung sucht den Ownersatz in der Set-Occurrence des Set *setname*, die den CRS enthält. Wenn der CRS bereits der Ownersatz ist, wird nur die Currency-Tabelle aktualisiert, da der Satz nicht neu in der Datenbank gesucht werden muss.

5.2.8 Sequenzieller Zugriff auf Realm-Ebene (FIND/FETCH-4)



Mit dieser Zugriffsart wählen Sie einen Satz auf Grund der Position aus, die er innerhalb der logischen Reihenfolge der Sätze in der Auswahlmenge einnimmt.
Die Auswahlmenge kann sein:

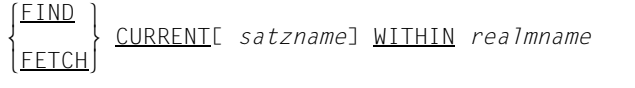
– bei *satzname* alle Sätze, die im Realm *realmname* liegen und zur Satzart *satzname* gehören. Der Realm-Name muss im Schema in der WITHIN-Klausel dieser Satzart genannt sein.

– bei RECORD alle Sätze des Realm *realmname*.

Die Reihenfolge der Sätze ist in beiden Auswahlmengen nach aufsteigenden Database-Key-Werten bestimmt.

Aus der Auswahlmenge können Sie den letzten oder den ersten Satz, den Nachfolger oder den Vorgänger des CRA oder den Satz auswählen, dessen Position einem anzugebenden Zahlenwert entspricht.

5.2.9 Zugriff auf den CRA (FIND/FETCH-5)



Hiermit setzen Sie die Currency-Information auf einen früheren Stand zurück. Der CRA des Realm *realmname* wird wieder zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle bzw. dort, wo Sie das Aktualisieren nicht mit RETAINING unterdrücken. Wenn Sie *satzname* angeben, greift UDS/SQL nur dann auf den CRA zu, wenn er zur angegebenen Satzart gehört.

5.2.10 CRU komplett oder teilweise in die UWA transportieren (GET)

```
GET [{satzname              }]
     {satzelementname,...   }
```

Diese Anweisung macht den zuletzt aus der Datenbank ausgewählten Satz - dieser Satz wird immer zum CRU - im Anwenderprogramm verfügbar, indem sie ihn komplett oder teilweise in die UWA transportiert.
Wenn Sie den CRU nicht komplett haben wollen, geben Sie eine Auswahl von Satzelementen an.
Die Funktion ist, soweit sie den CRU komplett in die UWA transportiert, in die FETCH-Anweisung integriert. In diesem Fall gilt:

```
FIND + GET = FETCH
```

5.2.11 Database-Key-Werte wiedergewinnen (ACCEPT-1)

```
ACCEPT feldname-1 FROM[ {satzname  } ] CURRENCY
                       {realmname }
                       {setname   }
```

Dieses Format überträgt einen Database-Key-Wert aus der Currency-Tabelle nach *feldname-1*.

Folgender Database-Key-Wert wird nach *feldname-1* übertragen:

– bei *satzname*: Database-Key-Wert des CRR

– bei *realmname*: Database-Key-Wert des CRA

– bei *setname*: Database-Key-Wert des CRS

Geben Sie keinen der Namen an, erhalten Sie den Database-Key-Wert des CRU.

Das Feld *feldname-1* muss mit USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG definiert sein.
Wenn der Database-Key-Wert aus der Currency-Tabelle eine REC-REF > 254 und/oder eine RSQ > 2^{24}-1 enthält, muss *feldname-1* mit USAGE IS DATABASE-KEY-LONG definiert sein. Andernfalls wird der Wert 0 in *feldname-1* abgeliefert und der DATABASE-STATUS 15 102 ausgegeben.

Wenn der angegebene Current Record nicht bekannt ist, wird ebenfalls der Wert 0 in *feldname-1* abgeliefert und der DATABASE-STATUS 15 102 ausgegeben.

5.2.12 Realm-Namen wiedergewinnen (ACCEPT-2)

```
ACCEPT feldname-2 FROM[ { satzname | setname | feldname-3 }] REALM-NAME
```

Dieses Format überträgt den Namen des Realm, auf den sich die Currency-Tabelle bezieht, nach *feldname-2*.

Folgender Realm-Name wird nach *feldname-2* übertragen:

- bei *satzname*: Name des Realm, zu dem der CRR gehört
- bei *setname*: Name des Realm, zu dem der CRS gehört
- bei *feldname-3*: Name des Realm, zu dem der Satz gehört, dessen Database-Key-Wert in *feldname-3* steht

Das Feld *feldname-3* muss mit USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG definiert sein.
Wenn Sie nach einem Satz suchen wollen, dessen Database-Key-Wert eine REC-REF > 254 und/oder eine RSQ > 2^{24}-1 enthält , muss *feldname-3* mit USAGE IS DATABASE-KEY-LONG definiert sein.

Wenn Sie weder *satzname*, *setname* noch *feldname-3* angeben, erhalten Sie den Namen des Realm, zu dem der CRU gehört.

5.3 Daten ändern

Daten können Sie wie folgt ändern:

- neue Sätze einspeichern und ggf. in Set-Occurrences einhängen (STORE)
- Sätze in Set-Occurrences einhängen (CONNECT, MODIFY)
- Sätze aus Set-Occurrences lösen (DISCONNECT, MODIFY)
- Feldinhalte verändern (MODIFY)
- Sätze und Setbeziehungen löschen (ERASE)

5.3.1 Satz in die Datenbank speichern und ggf. in Set-Occurrences einhängen (STORE)

```
STORE satzname[ RETAINING CURRENCY FOR{ MULTIPLE                                  }]
                                      {[ REALM][ RECORD][ {SETS          }]      }
                                                         {setname,...    }       }
```

STORE überträgt alle Feldinhalte der Satzart *satzname* aus der UWA in die Datenbank. Die Felder müssen Sie zuvor in der UWA mit den gewünschten Werten versorgen. Felder der Satzart, die nicht im Satzbereich der UWA enthalten sind, füllt UDS/SQL beim Einspeichern automatisch mit binären Nullen. (Letzteres ist nicht der Fall, wenn Sie die Sätze mit der CALL-DML komprimiert speichern; siehe STORE2 und STOR2L auf Seite 258.) STORE macht den Satz zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle.

In jedem Set, für den die Satzart im Schema als Owner definiert ist, wird der eingespeicherte Satz Owner einer neuen leeren Set-Occurrence. Wurde mit der SSL die Set-Occurrence-Population für den Set größer als 0 definiert, so werden ebenfalls die zugehörigen Tabellen angelegt.
Der Satz wird außerdem in alle Sets eingehängt, für die die Satzart im Schema als AUTOMATIC Member definiert ist. In diesem Fall müssen Sie zuvor die Set-Occurrences bestimmen, die den Satz aufnehmen sollen, und ggf. die Einfügungsorte innerhalb der Set-Occurrences.
Ist die Satzart laut Schema auf mehrere Realms verteilt, müssen Sie ebenfalls zuvor das zugehörige AREA-ID-Feld mit dem Namen des gewünschten Realms versorgen (siehe Handbuch „Entwerfen und Definieren“, WITHIN-Klausel), wenn die Satzart nicht Membersatzart einer verteilbaren Liste ist. Beim Abspeichern eines Membersatzes einer verteilbaren Liste wird die AREA_ID ignoriert. Freier Platz für neu anzulegende Stufe-0-Seiten der verteilbaren Liste wird im aktuell bevorzugten Realm ("Preferred-Realm") gesucht.

5.3.2 Satz in eine Set-Occurrence einhängen (CONNECT)

```
CONNECT[ satzname] TO { setname-1,... }
                      { ALL           }

      [RETAINING CURRENCY FOR { setname-2,... }]
                              { SETS          }
```

Die CONNECT-Anweisung wirkt auf den CRU.
Haben Sie eine Satzart als MANUAL Member in einem Set definiert, können Sie einen Satz dieser Satzart nur mit einer CONNECT-Anweisung in diesen Set einfügen. Der Satz muss schon in die Datenbank gespeichert sein.

Die CONNECT-Anweisung wirkt auf folgende Arten der Set-Mitgliedschaft von Membersätzen:

- MANDATORY MANUAL
  Ein Satz wird als MANUAL Member eingefügt (da er ein MANDATORY Membersatz ist, kann er nicht mit DISCONNECT ausgehängt, aber in eine andere Set-Occurrence umgehängt oder mit ERASE gelöscht werden).
- OPTIONAL AUTOMATIC
  Ein Satz, der schon einmal Membersatz einer Occurrence des Set war, wird wieder eingefügt (dieser AUTOMATIC Membersatz wurde mit STORE eingefügt und konnte, weil er OPTIONAL Membersatz ist, mit DISCONNECT wieder entfernt werden).
- OPTIONAL MANUAL
  Ein Satz wird zum ersten Mal nach einer STORE-Anweisung (MANUAL Membersatz) eingefügt oder als OPTIONAL Membersatz nach einer DISCONNECT-Anweisung wieder eingefügt.

Mit *satzname* nennen Sie eine Satzart mit einer der 3 Arten der Set-Mitgliedschaft in *setname-1*,.... Der CRU muss zu den Sätzen der angegebenen Satzart gehören.
Geben Sie keinen Satznamen an, so wird die Satzart aus der Beschreibung des ersten angegebenen Set ermittelt.

Die Set-Occurrence wird über den jeweiligen CRS ausgewählt. Nach der Ausführung des CONNECT ist der CRU der CRS jedes Set, in den er eingefügt wurde.
Wenn Sie mit RETAINING arbeiten, wird die Änderung der Informationen in der Currency-Tabelle für *setname-2*,... unterdrückt. Geben Sie SETS an, bleibt die Currency-Information aller betroffenen Sets unverändert.

5.3.3 Bestehende Set-Beziehungen lösen (DISCONNECT)

Satz aus Set-Occurrences entfernen

```
DISCONNECT[ satzname] FROM { setname,... }
                           { ALL        }
```

Diese DISCONNECT-Anweisung wirkt auf den CRU. Sie löst den CRU aus den Set-Occurrences aller angegebenen Sets. Die Satzart muss in diesen Sets wahlfreie Membersatzart sein (d.h. Set-Mitgliedschaft OPTIONAL MANUAL oder OPTIONAL AUTOMATIC).
Die Satzart von *satzname* muss mit der des CRU übereinstimmen.

Geben Sie keinen Satznamen an, so wird die Satzart aus der Beschreibung des ersten angegebenen Set ermittelt.

Die DISCONNECT-Anweisung verändert die Informationen der Currency-Tabelle nicht.

Alle Membersätze aus dynamischen Sets entfernen

```
DISCONNECT ALL FROM setname,...
```

Dieses Format wirkt auf alle Sätze von *setname,...*.
Wenn Sie alle Membersätze aus einem dynamischen Set entfernen wollen, verwenden Sie dieses Format der DISCONNECT-Anweisung.

setname,... müssen dynamische Sets sein.

Sie dürfen bei diesem Format auch den Ergebnis-Set einer FIND-7-Anweisung angeben.
Die DISCONNECT-Anweisung verändert die Informationen in der Currency-Tabelle nicht.

5.3.4 CRU ändern oder umhängen (MODIFY)

```
MODIFY { satzname                }[ { INCLUDING } { ALL                } MEMBERSHIP]
       { satzelementname,...     }  { ONLY      } { setname-1,...     }

      [RETAINING CURRENCY FOR { SETS              }]
                              { setname-2,...     }
```

MODIFY kann Feldinhalte des CRU ändern oder den CRU von einer Set-Occurrence in eine andere Set-Occurrence desselben Set umhängen.

Wenn Sie nur die erstgenannte Funktion wünschen, dürfen Sie weder ONLY noch INCLUDING angeben.

Bei ONLY führt UDS/SQL nur die zweite Funktion aus.
Bei INCLUDING führt UDS/SQL beide Funktionen aus.
Die Funktionen sind im Folgenden getrennt beschrieben.

Feldinhalte des CRU ändern

```
MODIFY { satzname                }[ RETAINING CURRENCY FOR { SETS          }]
       { satzelementname,...     }                        { setname,...     }
```

UDS/SQL überträgt entweder alle Feldinhalte der Satzart *satzname* oder die der angegebenen Satzelemente aus der UWA auf den CRU. Die zu ändernden Felder müssen Sie zuvor mit den gewünschten Inhalten in der UWA versorgen.

Wenn Sie durch MODIFY Schlüsselwerte ändern, aktualisiert UDS/SQL automatisch alle zugehörigen Zugriffspfade wie Hashbereiche und Tabellen und die DBTT. Insbesondere kann sich also die Reihenfolge von Sätzen innerhalb einer Set-Occurrence ändern.
Den Database-Key-Wert eines Satzes können Sie nicht verändern, selbst dann nicht, wenn Sie das zugehörige Database-Key-Feld mit einem neuen Wert überschreiben.

CRU in andere Set-Occurrence umhängen

```
MODIFY satzname ONLY { ALL          } MEMBERSHIP
                     { setname,...  }

      [RETAINING CURRENCY FOR { SETS         }]
                              { setname,...  }
```

In den Sets, in denen der CRU bereits Mitglied einer Set-Occurrence ist, kann der CRU einem anderen Ownersatz zugeordnet werden. Sie können den CRU aus allen seinen Set-Occurrences lösen und in eine andere Set-Occurrence des jeweiligen Set einhängen oder eine Auswahl von Sets nennen, in denen dies geschehen soll.

In jedem Set, in dem Sie den CRU umhängen wollen, müssen Sie die Set-Occurrence und ggf. auch die Position innerhalb der Set-Occurrence bestimmen, die den CRU aufnehmen soll.

5.3.5 Sätze und deren Set-Beziehungen löschen (ERASE)

```
ERASE satzname [{ PERMANENT } MEMBERS]
               { SELECTIVE }
               { ALL       }
```

Die ERASE-Anweisung bezieht sich auf den CRU. Die Satzart von *satzname* muss mit der Satzart des CRU übereinstimmen.

Die ERASE-Anweisung entfernt den CRU aus allen Set-Occurrences, in denen er Membersatz ist, und löscht ihn. Nach dem Löschen ist der Speicherplatz wieder frei. Der Database-Key-Wert ist erst nach dem Beenden der Transaktion wieder frei.

Löschen Sie einen Satz, müssen Sie gleichzeitig seine Beziehung zu anderen Sätzen beachten.

Der Eintrag zum CRU in der Currency-Tabelle wird gelöscht.
Die anderen Einträge in der Currency-Tabelle werden als gelöscht gekennzeichnet. Die Möglichkeit, sequenziell zu suchen (FIND/FETCH NEXT), bleibt erhalten.

Alle Satzarten und Set-Beziehungen, die Sie durch die ERASE-Anweisung direkt oder indirekt angesprochen haben, müssen im Subschema enthalten sein. Alle Realms, die in der WITHIN-Klausel der betroffenen Satzarten genannt werden, müssen im Subschema vorhanden sein.

5.3.6 Zusammenhang zwischen Art der Set-Mitgliedschaft und Anweisungen zum Ändern von Daten

Art der Set-Mitglied-schaft	**Set-Mitgliedschaft wird hergestellt durch**		**Set-Mitgliedschaft wird aufgelöst durch**		**Wechsel zwischen Set-Occurrences**
	STORE	CONNECT	ERASE	DISCONNECT	MODIFY
MANDATORY AUTOMATIC	JA	NEIN	JA	NEIN	JA
MANDATORY MANUAL	NEIN	JA	JA	NEIN	JA
OPTIONAL AUTOMATIC	JA	JA	JA	JA	JA
OPTIONAL MANUAL	NEIN	JA	JA	JA	JA

Tabelle 10: Kombination von Art der Set-Mitgliedschaft und Anweisungen zum Ändern von Daten

5.4 Satzschutz

Für Seiten, die weder durch den Schutz auf Realm-Ebene (mit USAGE-MODE EXCLUSIVE oder PROTECTED RETRIEVAL) noch durch den Schutz auf Seitenebene (automatischer Schutzmechanismus für geänderte Seiten) vor einer Benutzung durch andere Transaktionen geschützt sind, gibt es die Möglichkeit des Datenschutzes auf Satzebene:

– erweiterten Satzschutz einschalten (KEEP)
– erweiterten Satzschutz ausschalten (FREE)

Immer vorhanden ist der Schutz der Seite (teilhabendes Sperren), in der der CRU liegt.

Wollen Sie einen Satz über diesen Zeitraum hinaus schützen, müssen Sie den Satzschutz mit KEEP erweitern.

5.4.1 Erweiterten Satzschutz einschalten (KEEP)

```
KEEP
```

Diese Anweisung schützt den CRU vor dem Zugriff einer anderen Transaktion, auch für den Zeitraum, in dem er nicht mehr CRU ist. Dieser KEEP-Zustand kann nur durch FINISH (Beenden der Transaktion) oder FREE aufgehoben werden.

Wenn der Realm, zu dem der Satz gehört, mit USAGE-MODE UPDATE eröffnet ist, erzeugt die KEEP-Anweisung das exklusive Zugriffsrecht auf die Seite, in der der Satz (CRU) liegt. Informationen aus dieser Seite sind in diesem Fall nur noch dieser Transaktion zugänglich.

Wenn der Realm, zu dem der Satz gehört, mit USAGE-MODE RETRIEVAL eröffnet ist, legt die KEEP-Anweisung teilhabendes Sperren (siehe Seite 34) für die Seite fest, in der der Satz (CRU) liegt. Andere Transaktionen können in diesem Fall nur noch lesend auf die Seite zugreifen.

Dieser Seitenzugriffsschutz bleibt bis zum nächsten FREE bzw. bis zum Transaktionsende bestehen.

5.4.2 Erweiterten Satzschutz ausschalten (FREE)

```
FREE[ ALL]
```

Mit dieser Anweisung heben Sie den KEEP-Zustand auf.

Wenn Sie ohne ALL-Angabe arbeiten, heben Sie den KEEP-Zustand nur für den CRU auf. Ohne ALL-Angabe können Sie nur arbeiten, wenn Sie den durch KEEP geschützten Satz wieder zum CRU gemacht haben.

Wenn Sie mit der ALL-Angabe arbeiten, geben Sie alle die Sätze für andere Transaktionen frei, die mit KEEP geschützt waren. Der normale CRU-Satzschutz bleibt aber in beiden Fällen erhalten.

5.5 Im Programm Set-Mitgliedschaften prüfen (IF)

5.5.1 Set-Mitgliedschaft des CRU prüfen

```
IF[ NOT][ setname] {OWNER  } {anweisung-1  }[ ELSE {anweisung-2  }].
                   {MEMBER } {             }       {             }
                   {TENANT } {NEXT SENTENCE}       {NEXT SENTENCE}
```

Im Einzelnen prüfen Sie:

– mit OWNER, ob der CRU Membersätze besitzt

– mit MEMBER, ob der CRU Membersatz in einer Set-Occurrence ist

– mit TENANT, ob der CRU Ownersatz einer nicht leeren Set-Occurrence oder Membersatz in einer Set-Occurrence ist

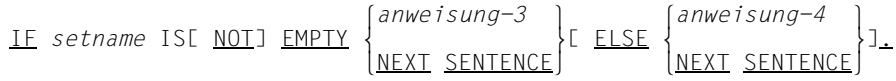
Geben Sie *setname* an, wird nur dieser Set geprüft; die Angabe eines dynamischen Set ist nicht erlaubt. Geben Sie *setname* nicht an, werden alle Sets des Subschemas geprüft, in denen die Satzart des CRU Owner- oder Membersatzart ist.

Das Ergebnis der Prüfung ist „WAHR“, wenn die Bedingung erfüllt ist. In diesem Fall wird nach *anweisung-1* oder NEXT SENTENCE verzweigt.
Ist das Ergebnis „UNWAHR“, wird nach *anweisung-2* oder NEXT SENTENCE verzweigt.

Bei NEXT SENTENCE wird auf eine Anweisung verzweigt, die der IF-Anweisung folgt.

Tritt ein Datenbanksonderzustand ein (siehe Abschnitt „Datenbanksonderzustände“ auf Seite 117), während eine IF-Anweisung ausgeführt wird, wird „UNWAHR“ angenommen und nach *anweisung-2* oder NEXT SENTENCE verzweigt.

5.5.2 Set-Occurrence auf Membersätze prüfen



Mit *setname* wählen Sie über den CRS die Set-Occurrence aus. Ein dynamischer Set darf nicht angegeben werden.

Das Ergebnis der Prüfung ist „WAHR“, wenn die Bedingung erfüllt ist. In diesem Fall wird nach *anweisung-3* oder NEXT SENTENCE verzweigt.
Ist das Ergebnis „UNWAHR“, wird nach *anweisung-4* oder NEXT SENTENCE verzweigt.

Tritt ein Datenbanksonderzustand ein (siehe Abschnitt „Datenbanksonderzustände“ auf Seite 117), während eine IF-Anweisung ausgeführt wird, wird „UNWAHR“ angenommen und nach *anweisung-4* oder NEXT SENTENCE verzweigt.

6 Anwenden der DML

Dieses Kapitel geht auf die Besonderheiten der COBOL-DML und CALL-DML ein und zeigt die verschiedenen Anwendungsmöglichkeiten.

Hier finden Sie eine Beschreibung der Funktionen, die COBOL-DML und CALL-DML unterscheiden. Die entsprechenden Formate finden Sie dann auf den Seiten 129 und 197.

6.1 Aufbau der COBOL-/CALL-DML-Programme

COBOL-DML-Programm

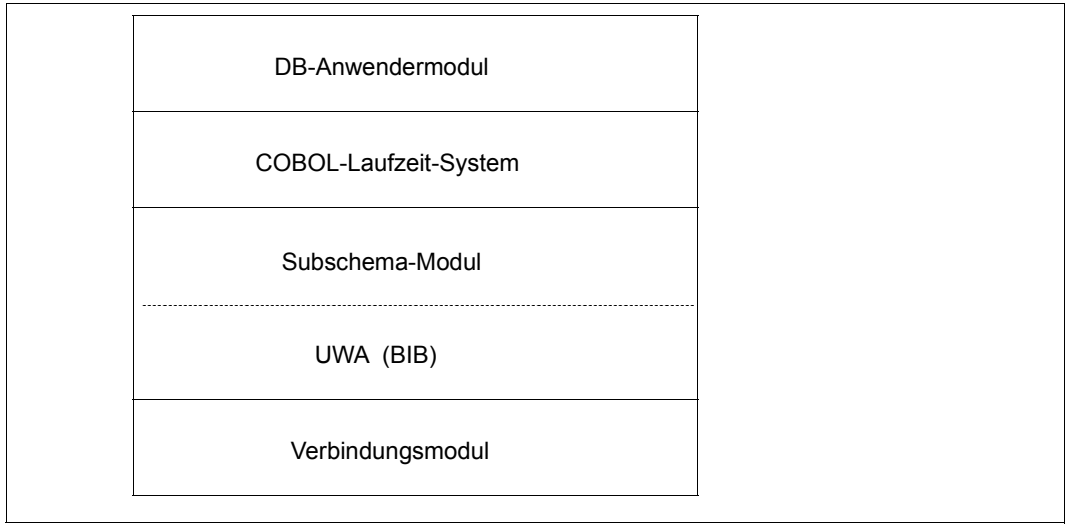


Bild 5: Aufbau eines COBOL-DML-Programms

CALL-DML-Programm

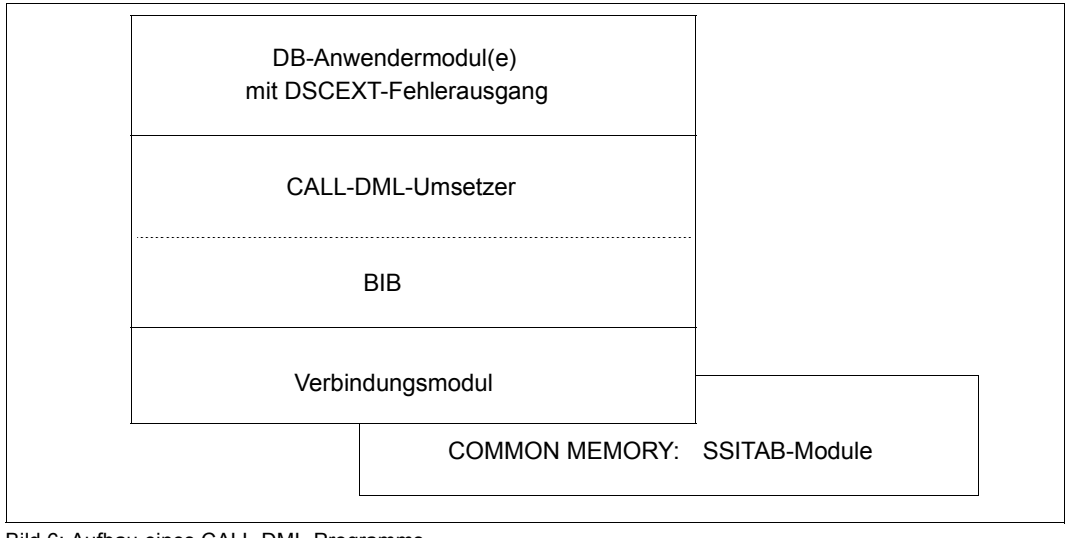


Bild 6: Aufbau eines CALL-DML-Programms

6.1.1 COBOL-DML

Die COBOL-DML kommt in drei Teilen eines COBOL-Programms vor:

– Identifikationsteil (ID DIVISION)
– Datenteil (DATA DIVISION)
– Prozedurteil (PROCEDURE DIVISION)

Die Anweisungen der DML gehören zum Funktionsumfang der ANSCOBOL-Compiler COBOL85 und COBOL2000.
Regeln und Formate zu den DML-Anweisungen stehen im Kapitel „Nachschlageteil der COBOL-DML“ auf Seite 129.

Programmerstellung

Die Voraussetzungen für ein ablauffähiges COBOL-DML-Programm sind:

– ein oder mehrere Anwendermodule
– ein oder mehrere Subschemata, die im COBOL-Subschema-Directory (COSSD) eingetragen sind

Programmaufbau

Sie können mit einer oder mehreren Datenbanken arbeiten.

Es folgen Beispiele für das Arbeiten mit einer Datenbank (Mono-DB-Betrieb) und mit mehreren Datenbanken (Multi-DB-Betrieb).

Mono-DB-Betrieb

Der Aufbau eines DB-Anwenderprogramms im Mono-DB-Betrieb ist aus folgendem Bild ersichtlich:

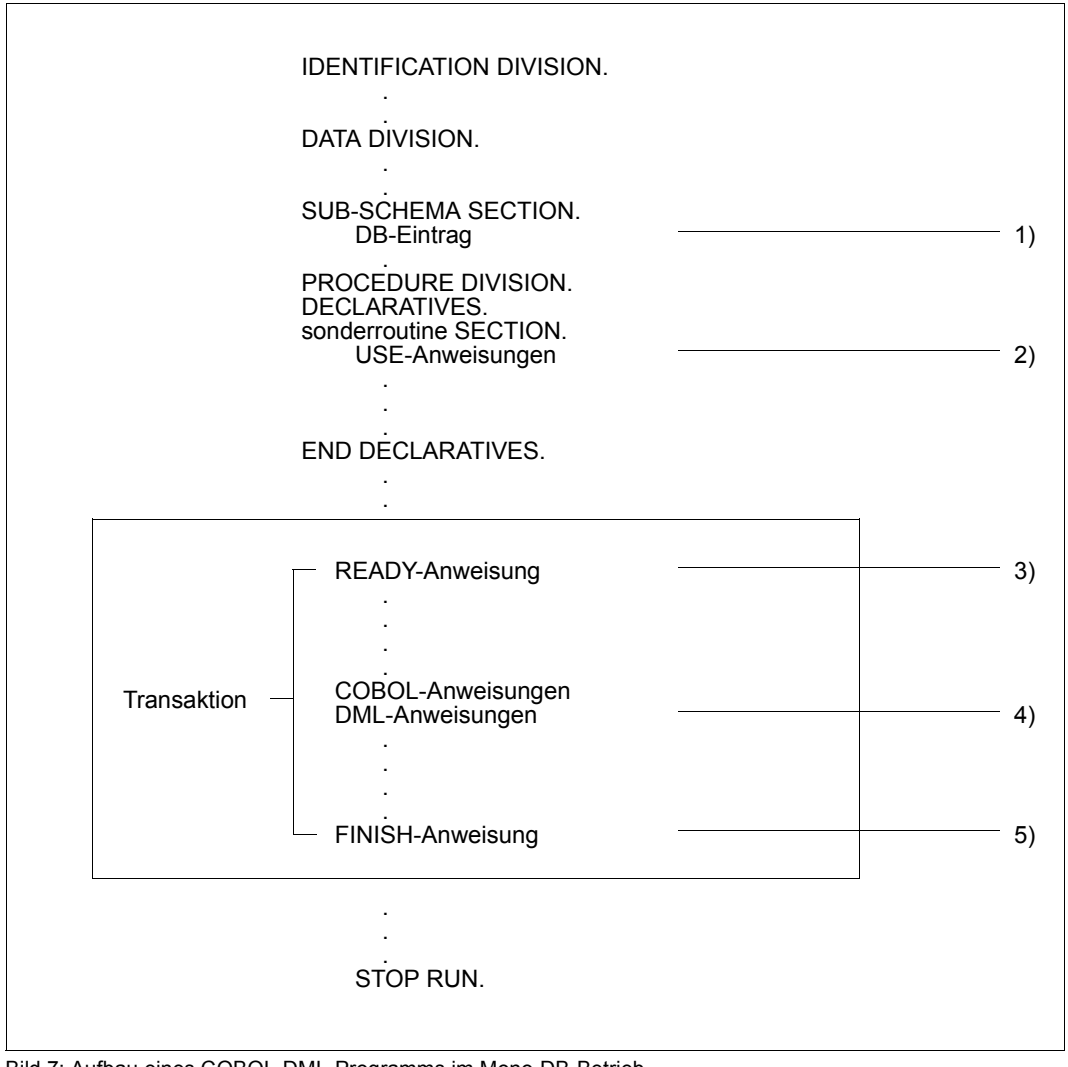


Bild 7: Aufbau eines COBOL-DML-Programms im Mono-DB-Betrieb

1) Der DB-Eintrag in der SUB-SCHEMA SECTION gibt den Namen des Subschemas an, auf das das Programm zugreifen soll. Der COBOL-Compiler benutzt den DB-Eintrag, um das Subschema im COSSD zu finden. Dieses Subschema übersetzt er zu einem Modul, das die UWA mit dem Benutzerverständigungsbereich enthält. Er legt das Subschemamodul im Common-Bereich des COBOL-Programms ab.

   Damit stehen alle das Subschema betreffenden Informationen bereit, die der DBH braucht, um auf die Datenbank zuzugreifen.

2) Mit Hilfe der USE-Anweisung können innerhalb der DECLARATIVES Befehlsfolgen definiert werden, die ausgeführt werden, wenn bei einer DML-Anweisung ein DB-Sonderzustand auftritt.

3) Mit der READY-Anweisung sichert sich der Anwender Zugriffsrechte zu den Realms für seine Verarbeitung. Sie stellt den Beginn einer Transaktion dar.

4) DML-Anweisungen ermöglichen den Zugriff auf die Datenbank (z.B. STORE, FIND).

5) FINISH beendet die Bearbeitung aller durch READY eröffneten Betriebsmittel (Realms und Datenseiten). Dabei wird eine Transaktion beendet.

Multi-DB-Betrieb

Bei COBOL-DML kann im DB-Eintrag der SUB-SCHEMA SECTION eines Anwendermoduls nur ein Subschema einer Datenbank angegeben werden. Soll auf mehrere Datenbanken zugegriffen werden, muss für jede Datenbank ein eigenes COBOL-Modul erstellt werden.

Beispiel 1

Zur Koordination von zwei DB-Anwendermodulen wird ein Steuermodul erstellt.

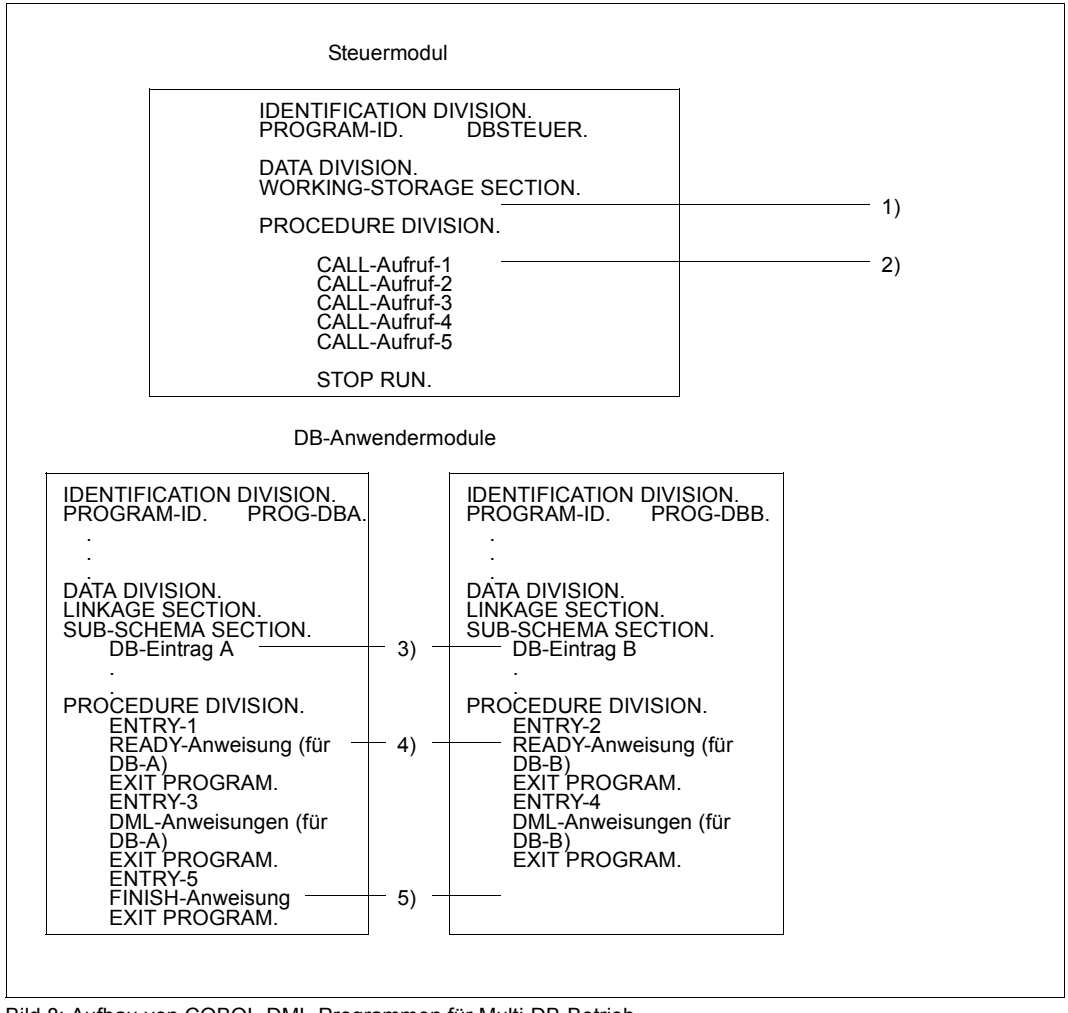


Bild 8: Aufbau von COBOL-DML-Programmen für Multi-DB-Betrieb

1) Das Steuermodul enthält keine SUB-SCHEMA SECTION.

2) Über CALL-Aufrufe werden die Einsprungstellen in die DB-Anwendermodule festgelegt.

3) In beiden Modulen wird jeweils ein Subschema einer Datenbank angegeben.

4) Die zuerst ausgeführte READY-Anweisung eröffnet eine Transaktion und gleichzeitig eine Verarbeitungskette. Jede weitere READY-Anweisung in einem anderen DB-Anwendermodul eröffnet jeweils eine Verarbeitungskette.

5) Im Multi-DB-Betrieb gibt es nur eine FINISH-Anweisung, die alle eröffneten Verarbeitungsketten und gleichzeitig die Transaktion schließt. Dabei ist es gleichgültig, von welchem DB-Anwendermodul diese FINISH-Anweisung gegeben wird (in diesem Beispiel von PROG-DBA).

Beispiel 2

Die Steuerung erfolgt in beiden Anwendermodulen.

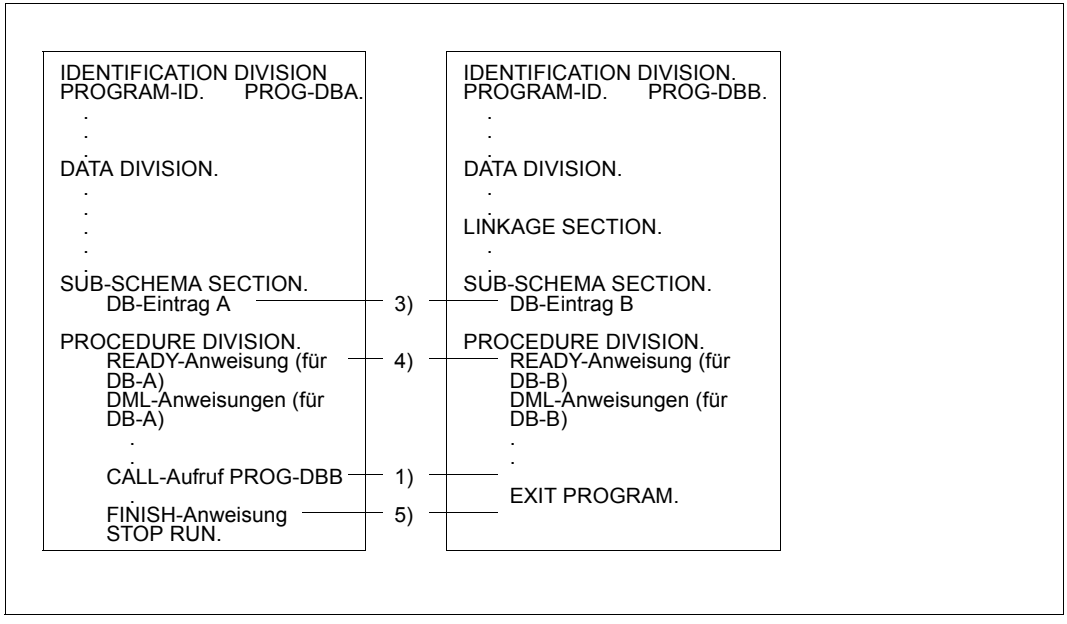


Bild 9: Wie Bild 8 ohne Steuermodul

1) Vom ersten DB-Anwendermodul wird das zweite Programm aufgerufen und behält die Steuerung, bis die EXIT PROGRAM-Anweisung die Rückkehr in das aufrufende Modul bewirkt.

3), 4), 5)

siehe Erläuterungen zum „Beispiel 1“ auf Seite 80

6.1.2 CALL-DML

Die CALL-DML stellt Ihnen die Leistungen der DML in einer Form zur Verfügung, die unabhängig ist von der verwendeten Programmiersprache.
Die Regeln der DML - soweit sie die Ausführung der Datenbankzugriffe betreffen - haben auch für die CALL-DML Gültigkeit. Die Formulierung der CALL-DML-Aufrufe unterscheidet sich jedoch grundsätzlich von denen der COBOL-DML. Die Anweisungen der COBOL-DML, die vom COBOL-Compiler verarbeitet werden, entfallen und werden ersetzt durch den Unterprogrammaufruf „CALL“ und eine Anzahl von Parametern. Diese Parameter (Funktionsname, Funktionsauswahl, Zusatzwahl usw.) beschreiben den Aufruf und übergeben den Datenbanknamen und die Daten (Benutzerinformation, Satzbereich).

Diese Übergabe der Aufruf-Information bedingt einige funktionelle Unterschiede zu COBOL-DML: CALL-DML benötigt einen eigenen Umsetzer. Der Umsetzer der CALL-DML bringt die Datenbank-Aufrufe erst zur Laufzeit in die Form, die der Database Handler verarbeiten kann, und prüft sie auf Richtigkeit und Zulässigkeit.

Die Parameter der CALL-DML-Aufrufe können jedoch auch erst zur Laufzeit vom Anwender eingegeben oder aus einer Datei gelesen werden. Ein solches Verfahren wird zusätzlich durch die in COBOL-DML nicht enthaltene LOOKC-Funktion unterstützt, die einen Zugriff auf Strukturinformationen des Subschemas gestattet.

CALL-DML-Programme können mit openUTM ablaufen.

Bei der Übersetzung eines CALL-DML-Anwenderprogramms braucht das verwendete Subschema noch nicht bekannt zu sein. Es werden keine Funktionen angesprochen, die über den üblichen Umfang der Compiler hinausgehen.

Sie definieren die Übergabebereiche von Sätzen und Parametern selbst in Ihrem Programm.

Die Adressen der jeweils benötigten Übergabebereiche werden bei jedem CALL-DML-Aufruf übergeben; diese Bereiche können daher für jeden Aufruf neu gewählt oder beliebig wieder verwendet werden.

CALL-DML-Programm erstellen

Ein ablauffähiges CALL-DML-Programm enthält folgende Teile:

– ein oder mehrere Anwendermodule

– UDS/SQL-Verbindungsmodule

– pro Subschema ein SSITAB-Modul, das mit dem Dienstprogramm BCALLSI erstellt wurde (siehe Handbuch „Aufbauen und Umstrukturieren“).

Programmaufbau

Bei CALL-DML können Sie von einem Modul aus mit mehreren Datenbanken arbeiten (Multi-DB-Betrieb). Es folgt ein Beispiel für den Programmaufbau eines COBOL-Programms bei Multi-DB-Betrieb.

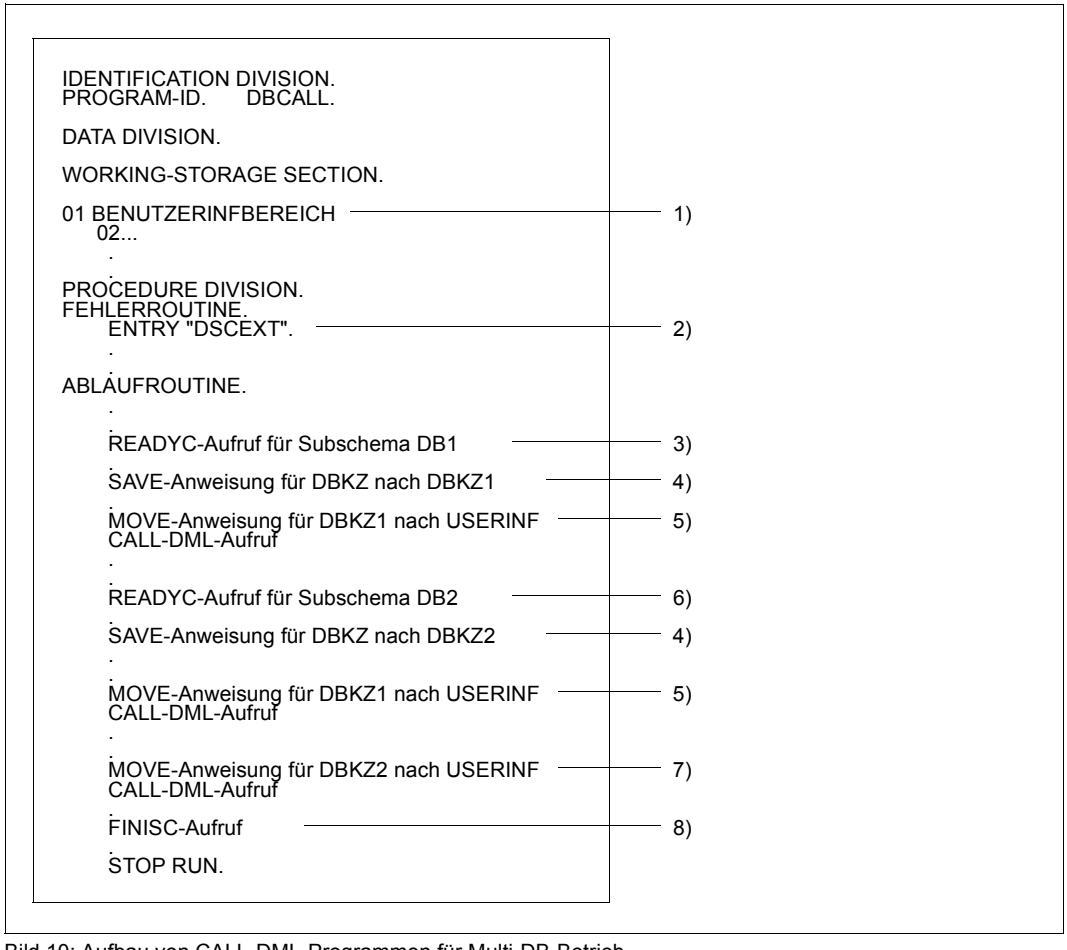


Bild 10: Aufbau von CALL-DML-Programmen für Multi-DB-Betrieb

1) Die Übergabebereiche von Sätzen und Parametern müssen Sie bei CALL-DML im Programm definieren. Den Parameterbereich Benutzerinformation müssen Sie nach einem fest vorgeschriebenen Format erstellen.

2) Das Programm muss eine Fehlerbehandlungsroutine mit dem ENTRY-Namen DSCEXT enthalten (siehe Abschnitt „Fehlerbehandlungsroutine DSCEXT der CALL-DML“ auf Seite 124).

3) Der CALL-Aufruf mit Funktionsname READYC eröffnet die Transaktion.
Der erste READYC-Aufruf eröffnet gleichzeitig die Verarbeitungskette für das angegebene Subschema der Datenbank DB1.
Dem Anwender wird im Benutzerinformationsbereich ein eindeutiges Datenbankkennzeichen übergeben.

4) Das Datenbankkennzeichen müssen Sie in einem Feld sicherstellen.

5) Bei jedem CALL-DML-Aufruf außer READYC muss das Datenbankkennzeichen übergeben werden, da die CALL-DML-Aufrufe der jeweiligen Datenbank über das Datenbankkennzeichen zugeordnet werden.

6) Der zweite READYC-Aufruf eröffnet den Zugriff auf die Datenbank DB2.

7) Erst durch die Übergabe des Datenbankkennzeichens DBKZ2 wird dem folgenden CALL-DML-Aufruf die Datenbank DB2 zugewiesen.

8) FINISC beendet die Bearbeitung aller durch READYC belegten Betriebsmittel. Dabei wird gleichzeitig die Transaktion beendet.

6.2 Besonderheiten der COBOL-DML

Schlüssel für die Benutzung des Subschemas angeben (PRIVACY)

```
[PRIVACY. PRIVACY KEY FOR COMPILE IS literal.]
```

Ist in dem Subschema, mit dem Sie arbeiten, ein Schloss für den Zugriff auf das Subschema definiert, nämlich die PRIVACY LOCK FOR COMPILE-Klausel, müssen Sie beim Übersetzen einen passenden Schlüssel, einen PRIVACY KEY, in Ihrem COBOL-Programm angeben. Die PRIVACY KEY-Klausel steht in der IDENTIFICATION DIVISION.

Subschema zuweisen und Verständigungsbereich einrichten (DB-Eintrag)

Im Datenteil des COBOL-Programms geben Sie den DB-Eintrag als Teil der SUBSCHEMA SECTION an.

```
DB subschemaname WITHIN schemaname.
```

Mit dem DB-Eintrag geben Sie an, welches Subschema Ihr COBOL-DML-Programm verwenden soll.

Dieser Eintrag wird sowohl bei der Übersetzung als auch bei der Ausführung des Programms ausgewertet.

Durch das Subschema wird der Satzbereich (RECORD AREA) festgelegt. Der Satzbereich darf nicht länger als 65 535 byte sein. Die Länge des Satzbereichs wird wesentlich durch die Länge der im zu Grunde liegenden Subschema enthaltenen Satzarten bestimmt. Der Satzbereich nimmt je einen Satz jeder Satzart des Subschemas auf und enthält außerdem die IMPLICITLY-DEFINED-DATA-NAMES, d.h. den Bereich für die implizit definierten Felder der Datenbank (z.B. AREA-ID-Felder der WITHIN-Klauseln des Schemas).

Die Nutzung von Satzbereichen bis 65 535 byte Länge ist mit COBOL2000 bzw. COBOL85 ab V2.2C21 möglich, sofern nicht bei der Übersetzung des Subschemas die Anweisung SUBSCHEMA FORM IS OLD angegeben wurde (siehe Handbuch „Aufbauen und Umstrukturieren“).

Bei der Übersetzung wird die User Work Area (UWA) generiert. Der Inhalt dieser UWA ist Bestandteil des Anwenderprogramms und abhängig vom aufgerufenen Subschema. Vor der Übersetzung des COBOL-DML-Programms mit dem COBOL-Compiler muss mit dem Linknamen DATABASE die zu bearbeitende Datenbank zugewiesen werden:
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=*dbname*

In der UWA sind folgende Bereiche immer generiert:

– SYSTEM-COMMUNICATION-LOCATIONS enthält die COBOL-Sonderregister (siehe Seite 88)

– Satzbereich (RECORD AREA)

– PRIVACY RECORD (enthält die Felder für die BPRIVACY-Schlösser):

  Eine Versorgung des PRIVACY-RECORD durch das COBOL-Programm ist nicht erforderlich; eventuell vorhandene Angaben werden von UDS/SQL ignoriert. Die PRIVACY-Daten werden außerhalb des COBOL-Programms durch openUTM bzw. BS2000 an UDS/SQL übergeben.

  In Ihrem COBOL-Programm können Sie alle Felder mit den in der UWA generierten Namen ansprechen.

Wenn mehrere Module zu einem Programm zusammengebunden werden, kann jedes Modul einen DB-Eintrag enthalten.
Der DB-Eintrag führt zur Ausführung einer Startroutine, die den Database Handler aufruft. Zur Laufzeit des Anwenderprogramms wird bei der Ausführung der READY-Anweisung der Name des Subschemas aus dem DB-Eintrag entnommen und dem Database Handler übergeben, der dann eine entsprechende SSIA, die Subschema Information Area lädt. Die SSIA ist die vom DBH benutzte interne Darstellung eines Subschemas. Alle SSIAs sind im Database Directory (DBDIR) gespeichert.

Neben den SSIAs existiert pro Datenbank eine SIA, Schema Information Area. Der DBH kann jederzeit auf die SIA zugreifen, um die für die Ausführung von DML-Anweisungen notwendigen Schema-Informationen zu erhalten.

COBOL-Sonderregister

Bei den COBOL-Sonderregistern handelt es sich um vom COBOL-Compiler erzeugte Speicherbereiche. Die Sonderregister werden pro DB-Eintrag nur einmal angelegt. Sie können in Ihrem COBOL-DML-Programm auf die Werte der Sonderregister zugreifen.

Die Register haben folgende Formate:

```
DATABASE-REALM-NAME                  PIC X(30)
DATABASE-RECORD-NAME                 PIC X(30)
DATABASE-SET-NAME                    PIC X(30)
DATABASE-STATUS                     PIC 9(5)
```

Der DBH ändert die Werte der COBOL-Sonderregister nach DML-Anweisungen und trägt die aktuellen Werte ein (siehe Abschnitt „Verwendung der Sonderregister (COBOL-DML) bzw. des Benutzerinformationsbereichs (CALL-DML)“ auf Seite 117).

Database-Key-Wert übertragen (SET)

SET *feldname-1*,... TO *feldname-2*

Mit dieser Anweisung übertragen Sie einen Database-Key-Wert in ein oder mehrere Felder. Alle verwendeten Felder müssen mit USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG definiert sein. Der Inhalt von *feldname-2* wird nach *feldname-1*,... übertragen.

Näheres zur Übertragung von Database-Key-Werten mit der SET-Anweisung finden Sie im Kapitel „Nachschlageteil der COBOL-DML“ unter „SET“ auf Seite 188.

Fehlerbehandlungsroutine beschreiben (USE)

```
USE FOR DATABASE-EXCEPTION[ ON { OTHER          }].
                               { literal-1,...  }
```

Mit der USE-Anweisung leiten Sie Behandlungsroutinen für Datenbanksonderzustände ein (siehe Abschnitt „Datenbanksonderzustände“ auf Seite 117). Diese Routinen dürfen Sie nur in den DECLARATIVES eines COBOL-DML-Programms angeben.

Datenbanksonderzustände treten nicht nur nach Fehlern auf. Der DBH versorgt nach jeder COBOL-DML-Anweisung die COBOL-Sonderregister. Sie können durch das Überprüfen des DATABASE-STATUS mit der USE-Anweisung auch feststellen, ob z.B. von einem Satz ein Duplikat gefunden werden konnte oder nicht. Sie können Routinen programmieren, je nach Art des Datenbanksonderzustandes (siehe Tabelle 15 auf Seite 119).

Sie können unterschiedliche USE-Anweisungen definieren und damit Ihr Programm steuern. Dafür gibt es zwei Möglichkeiten:

– Datenbanksonderzustände abfragen, die auftreten, wenn eine DML-Anweisung nicht erfolgreich beendet werden konnte,

– einen bestimmten Datenbanksonderzustand als Schalter für Ihre Programmlogik benutzen.

Die DECLARATIVES eines COBOL-DML-Programms werden nach jeder DML-Anweisung durchlaufen, die einen Datenbanksonderzustand erzeugt hat, und verringern den Programmieraufwand.

Zuweisung der COSSD-Datei für die Übersetzung eines COBOL-DML-Programms

Bei der Kompilierung eines COBOL-DML-Anwenderprogramms muss der COBOL-Compiler die COSSD-Datei der betroffenen Datenbank lesen. Deshalb müssen Sie dem COBOL-Compiler die COSSD-Datei auf eine der folgenden Arten zuweisen:

- Zuweisung über LINK-NAME=UDSCOSSD
- Zuweisung über LINK-NAME=DATABASE

Zuweisung über LINK-NAME=UDSCOSSD

Dieses Verfahren wird nur vom COBOL2000-Compiler ab Version V1.4 unterstützt.

Dem COBOL-Compiler wird die COSSD-Datei explizit zugewiesen mit dem Kommando

```
/ADD-FILE-LINK      LINK-NAME=UDSCOSSD, -
/                   FILE-NAME=[:catid:][$userid.]dbname.COSSD
```

Dabei sind `:catid:` und `$userid` die Katalogkennung und Benutzerkennung, unter der die COSSD-Datei katalogisiert ist. Ohne die Angabe `:catid:` bzw. `$userid` wird der Dateiname nach den Standardregeln des BS2000 komplettiert.

Die COSSD-Datei muss unter dem im Kommando angegeben Namen katalogisiert sein, da im Fehlerfall nicht nach einer COSSD-Datei an anderer Stelle gesucht wird.

Dieses Verfahren ist zwingend erforderlich, wenn in allen Katalogen, die lokal von der Benutzerkennung aus zugreifbar sind, mehrere COSSD-Dateien mit dem entsprechenden Datenbanknamen existieren.

Eine evtl. vorhandene UDS/SQL-Pubset-Deklaration wird **nicht** berücksichtigt.

Beispiel für eine Kommandofolge:

```
/ADD-FILE-LINK LINK-NAME=UDSCOSSD,FILE-NAME=dbname.COSSD
/START-COBOL2000-COMPILER -
/SOURCE=cobolsource, -
/COMPILER-ACTION=MODULE-GENERATION(MODULE-FORMAT=LLM), -
/MODULE-OUTPUT=*LIBRARY(LIBRARY=bibliothek-1,ELEMENT=element)
```

Zuweisung über LINK-NAME=DATABASE

Dieses Verfahren wird von allen COBOL2000- und COBOL85-Compilern unterstützt.

Dem COBOL-Compiler wird der Datenbankname mitgeteilt mit dem Kommando

```
/SET-FILE-LINK      LINK-NAME=DATABASE, -
/                   FILE-NAME=[:catid:][$userid.]dbname
```

Die Angabe einer `:catid:` beim Kommando SET-FILE-LINK wird ignoriert. Der COBOL-Compiler sucht dann eine COSSD-Datei mit dem Namen `dbname.COSSD`
in allen Katalogen, die lokal von derjenigen Benutzerkennung aus zugreifbar sind, die beim Kommando SET-FILE-LINK explizit angegeben wurde oder vom BS2000 ergänzt wurde.

Dieses Verfahren kann nur verwendet werden, wenn in allen Katalogen, die lokal von der Benutzerkennung aus zugreifbar sind, nur eine COSSD-Datei mit dem entsprechenden Datenbanknamen existiert.

Eine evtl. vorhandene UDS/SQL-Pubset-Deklaration wird **nicht** berücksichtigt.

Falls gleichzeitig eine Zuweisung für LINK=UDSCOSSD vorliegt, wird nur das Verfahren für LINK=UDSCOSSD angewandt.

Beispiel für eine Kommandofolge:

```
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname
/START-COBOL2000-COMPILER -
/SOURCE=cobolsource, -
/COMPILER-ACTION=MODULE-GENERATION(MODULE-FORMAT=LLM), -
/MODULE-OUTPUT=*LIBRARY(LIBRARY=bibliothek-1,ELEMENT=element)
```

Fehlerbehandlung

Falls der COBOL-Compiler die COSSD-Datei nicht bearbeiten kann, gibt er eine Meldung ERROR ACCESSING SUB-SCHEMA aus.

Diese Meldung wird auch ausgegeben, wenn bei der Zuweisung über LINK-NAME=DATABASE die Datei in unterschiedlichen Pubsets mehrfach vorliegt. Benutzen Sie in diesem Fall die spezifische Zuweisung der COSSD-Datei über LINK-NAME=UDSCOSSD.

6.3 Besonderheiten der CALL-DML

Umsetzung zur Ablaufzeit

Die CALL-DML-Aufrufe werden erst zur Laufzeit des Programms in die vom DBH benötigte Form umgesetzt. Sie werden auch erst dann auf Zulässigkeit geprüft. Daraus folgt, dass die CALL-DML viel flexibler eingesetzt werden kann, weil es auch möglich ist, die Aufrufe in Abhängigkeit von der Dateneingabe während des Programmlaufs zu generieren.

Verkürztes Satzformat

Das starre Subschema-Satzformat kann durch ein verkürztes Satzformat ersetzt werden. Es muss im Programm dann nur der Platz für die benötigten Felder bereitgestellt werden. Das verkürzte Satzformat ist möglich bei den Aufrufen GETC, MODIF2, STORE2 und STOR2L; es wird durch den Zusatzwahl-Parameter VAR angesprochen. Auf die Länge des Satzes in der Datenbank hat dies nur Einfluss, wenn der Satz mit COMPRESSION ALL definiert ist (siehe Handbuch „Entwerfen und Definieren“, Komprimierung).

Implizit definierte Datenbereiche (IMPLICITLY-DEFINED-DATA-NAMES)

In der Schema-DDL werden Feldnamen für Felder vereinbart, die zur Übergabe von Daten an den DBH, außerhalb der Sätze, bestimmt sind. Dazu zählen das Database-Key-Feld der Klausel LOCATION MODE DIRECT/DIRECT-LONG und das AREA-ID-Feld der WITHIN-Klausel (siehe Handbuch „Entwerfen und Definieren“).
Bei CALL-DML wird auf die vereinbarten Feldnamen nicht Bezug genommen, die entsprechenden Daten werden mit einem eigenen Parameter übergeben (Spezialparameter-2: implizit definierter Datenbereich bei FIND/FETCH und STORE).

Sets mit SET SELECTION THRU LOCATION MODE OF OWNER

Sets, die mit der SET OCCURRENCE SELECTION-Klausel THRU LOCATION MODE OF OWNER definiert sind, werden behandelt wie Sets mit der SET OCCURRENCE SELECTION-Klausel CURRENT OF SET.

Benutzerinformationsbereich

Zur Kommunikation mit dem DBH dient der Parameterbereich Benutzerinformation, der in einem fest vorgegebenen Format zu erstellen ist und bei jedem CALL-Aufruf anzugeben ist. Er enthält z.B. Felder für die Sonderregister, den Datenbank-Status, Übergabefelder für einen Database-Key-Wert und den Zähler für die TALLYING-Funktion des FIND7A/FTCH7A.

Variable Felder

Die CALL-DML gestattet bei Verwendung einer geeigneten Programmiersprache ein problemloses Arbeiten mit einem variablen Feld, d.h. mit der Speicherung von Sätzen in variabler Länge. Das Längenfeld des variablen Feldes muss vom Anwender selbst mit einem binären Wert versorgt werden.

Subschema-Informationen

Bei der CALL-DML darf der durch das Subschema festgelegte Satzbereich (RECA) maximal 65 535 byte lang sein. Die Länge des Satzbereichs wird wesentlich durch die Länge der im zu Grunde liegenden Subschema enthaltenen Satzarten bestimmt. Der Satzbereich nimmt je einen Satz jeder Satzart des Subschemas auf und enthält außerdem die IMPLICITLY-DEFINED-DATA-NAMES, d.h. den Bereich für die implizit definierten Felder der Datenbank (z.B. AREA-ID-Felder der WITHIN-Klauseln des Schemas).
Wenn bei der Übersetzung des Subschemas die Anweisung SUBSCHEMA FORM IS OLD angegeben wurde, darf der zugehörige Satzbereich maximal 61 328 byte lang sein.

Die Beschreibung des verwendeten Subschemas muss CALL-DML in Form spezieller Module zur Verfügung gestellt werden, die einen grundsätzlich von den Subschema-Modulen der COBOL-DML unterschiedlichen Aufbau haben. Diese Module werden als SSITAB-Module bezeichnet (Subschema Information Table). Sie werden von dem Dienstprogramm BCALLSI erzeugt (siehe Handbuch „Aufbauen und Umstrukturieren“).

Von einem CALL-DML-Anwendungsprogramm können verschiedene Subschemata aufgerufen werden. Subschemanamen müssen in den ersten 6 Bytes eindeutig sein, um die Unverwechselbarkeit der SSITAB-Module zu gewährleisten.

Es können über einen Feldnamen nur Elementarfelder angesprochen werden. Indizierte Felder können in der CALL-DML nicht benannt werden. Dies ist programmiertechnisch bedingt, bedeutet jedoch keine Einschränkung des Funktionsumfangs gegenüber COBOL-DML.

Struktur des Subschemas prüfen (LOOKC)

Wollen Sie in CALL-DML-Aufrufen die Parameterleiste ändern, benötigen Sie eine genaue Information über das Subschema.

Sie können folgende Informationen mit

```
CALL DML, LOOKC...
```

erhalten:

LOOKC nach	**Strukturinformationen über**
einem Namen	– ein oder mehrere Elemente – den Owner/Member eines Set
einem Realm	– Realms, in denen Sätze einer bestimmten Satzart gespeichert sein können
einer Satzart	– eine oder mehrere Satzarten – die Owner-/Membersatzart eines bestimmten Set
einem Set	– einen oder mehrere Sets – einen oder mehrere Sets, in denen eine bestimmte Satzart Owner- bzw. Membersatzart ist
einem Feld	– ein oder mehrere Felder einer bestimmten Satzart
einem Schlüssel	– einen oder mehrere Schlüssel eines bestimmten Set – einen oder mehrere Schlüssel eines Set, in denen ein bestimmtes Feld enthalten ist – einen oder mehrere Schlüssel, in denen ein bestimmtes Feld enthalten ist
den Feldern eines Schlüssels	– ein oder mehrere Felder eines bestimmten Schlüssels

Tabelle 11: Strukturinformationen mit LOOKC

6.4 UDS/SQL-TIAM-Anwenderprogramm binden, laden und starten

6.4.1 Grundlegende Aspekte

Prinzip des dynamischen Nachladens

In das UDS/SQL-Anwenderprogramm wird lediglich ein „versionsunabhängiges“ Verbindungsmodul (UDSLNKI, UDSLNKL oder UDSLNKA) zum Anschluss an den independent oder den linked-in DBH fest eingebunden (siehe auch Abschnitt „UDS/SQL-TIAM-Anwendungen binden“ auf Seite 100). Alle weiteren zum Ablauf einer UDS/SQL-Anwendung benötigten Verbindungsmodule des Produkts sowie die anwendungsspezifischen Datenmodule (SSITAB, PLITAB) werden vom DBL zum Ablaufzeitpunkt dynamisch nachgeladen. Dieses Konzept hat den Vorteil, dass UDS/SQL-Anwendungen zum Zeitpunkt eines UDS/SQL-Versionswechsels **nicht** neu gebunden werden müssen. Ein evtl. längerfristig notwendiges Neubinden kann zu einem späteren Zeitpunkt und sukzessiv für einzelne Anwendungen erfolgen.

Vor dem Start des Anwenderprogramms müssen die Bibliotheken, die die nachzuladenden Produkt- und Datenmodule enthalten, zur Verfügung stehen bzw. zugewiesen werden. Im Folgenden werden die verschiedenen Nachladestrategien (und damit Zuweisungstechniken) getrennt nach den Produkt- und Datenmodulen vorgestellt.

Nachladen der UDS/SQL-Produktmodule

Im Standardfall wird UDS/SQL mit IMON installiert und im Software Configuration Inventory (SCI) zentral verwaltet. Dabei können mehrere Produktversionen parallel installiert und mehrere UDS/SQL-Subsystemversionen vorgeladen sein. Für die Zuweisung der Produktbibliothek, aus der Module nachgeladen werden, sowie für die Auswahl der passenden Subsystemversion steht das BS2000-Kommando SELECT-PRODUCT-VERSION zur Verfügung:

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=product,VERSION=version,SCOPE=*TASK
```

product
Name der Liefereinheit (z.B. UDS-SQL oder UDS-D)

version

Die Versionsnummer sollte immer inklusive Freigabe- und Korrekturstand angegeben werden (z.B. 02.8A00). Die jeweils mit SELECT-PRODUCT-VERSION angegebene Versionsnummer muss nicht in jeder Anwenderprozedur fest vergeben werden, sondern kann über eine Jobvariable oder S-Variable auf einen zentral geführten Wert gesetzt werden.
Im Handbuch „Datenbankbetrieb“, Abschnitt „Mehrere UDS/SQL-Versionen parallel nutzen“ finden Sie Beispiele für Startprozeduren, in denen u.a. die mit SELECT-PRODUCT-VERSION angegebene Versionsnummer mit Hilfe von Jobvariablen variabel gestaltet wird.

Beim Aufruf des Anwenderprogramms, in das das versionsunabhängige Verbindungsmodul für den **independent** DBH eingebunden ist, wird mit SELECT-PRODUCT-VERSION die Version des nachzuladenden, versionsabhängigen Verbindungsmoduls UDSBCCON ausgewählt.
Beim Aufruf des Anwenderprogramms, in das das versionsunabhängige Verbindungsmodul für den **linked-in** DBH eingebunden ist, wird mit SELECT-PRODUCT-VERSION die Version des nachzuladenden, versionsabhängigen Verbindungsmoduls LCCONCT ausgewählt.
Das eigentliche UDS/SQL-Coding wird in der passenden Version zum Verbindungsmodul als vorgeladenes Subsystem genutzt bzw. aus der entsprechenden SYSLNK-Bibliothek des Produkts UDS/SQL nachgeladen.
Für dieses Nachladeverfahren wird in den Fehlermeldungen des Produkts die Abkürzung „SCI“ verwendet.

Der Einsatz des SELECT-PRODUCT-VERSION-Kommandos wird auch dann empfohlen, wenn über längere Zeiträume nur eine UDS/SQL-Version auf der Anlage genutzt wird.

Wenn SELECT-PRODUCT-VERSION nicht angegeben wird, wird beim ersten Nachladen im versionsunabhängigen Modul versucht, aus einer ggf. vorhandenen Privatinstallation nachzuladen (siehe Seite 97). Wenn keine Privatinstallation gefunden wird, wird die Standard-Version aus dem SCI nachgeladen.

UDS-D-Einsatz:

Bei Einsatz von UDS-D (nur mit dem independent DBH möglich) sollte mit einem zusätzlichen SELECT-PRODUCT-VERSION-Kommando die zur UDS/SQL-Version korrespondierende UDS-D-Version angegeben werden, wenn UDS-D als Subsystem genutzt wird (siehe hierzu ggf. die Freigabmitteilung). Auch hier wird empfohlen, SELECT-PRODUCT-VERSION auch dann zu verwenden, wenn über längere Zeiträume nur eine UDS-D-Version auf der Anlage genutzt wird.

Privatinstallation

Wenn das Produkt UDS/SQL nicht rechnerweit im SCI verwaltet wird, können die Module aus Bibliotheken einer Privatinstallation nachgeladen werden.

Die Bibliotheken, aus denen die Module des Produkts UDS/SQL und ggf. der Produkte UDS-D und UDSKDBS in der gewünschten Version nachgeladen werden sollen, werden explizit mit dem ADD-FILE-LINK-Kommando zugewiesen. Folgende Linknamen stehen für die Zuweisungen zur Verfügung:

$UDSLIB	Module des Produkts UDS/SQL (independent und linked-in DBH)
$UDSDLIB	Module des Produkts UDS-D (nur bei independent DBH)
$UDSKLIB	Module des Produkts KDBS (nur bei linked-in DBH)

Für dieses Verfahren wird in den Fehlermeldungen des Produkts die Abkürzung „$UL“ verwendet.

Aus Kompatibilitätsgründen wird auch weiterhin die Nutzung einer UDS.MODLIB mit ggf. TASKLIB-Zuweisung zum Nachladen der Module unterstützt, wenn die Produkte weder über SCI noch über die Linknamen $UDSLIB, $UDSDLIB und $UDSKLIB bereitgestellt sind. Für dieses Verfahren wird in den Fehlermeldungen des Produkts die Abkürzung „TSK“ verwendet.

Die SELECT-PRODUCT-VERSION-Kommandos für die Produkte UDS/SQL und ggf. UDS-D werden aus folgenden Gründen evtl. auch bei einer Privatinstallation benötigt:

Parallel zu einer privat installierten Produktversion können auch UDS/SQL- und UDS-D-Subsystemen vorgeladen sein. Mit dem SELECT-PRODUCT-VERSION-Kommando wird sichergestellt, dass die aus der Privatinstallation nachgeladenen Module eine Verbindung zur richtigen Subsystemversion herstellen.

Nachladen der anwendungsspezifischen Datenmodule SSITAB und PLITAB

Im Falle von CALL-DML-Anwenderprogrammen muss die Bibliothek bekannt sein, aus der zum Ablaufzeitpunkt die SSITAB-Module nachgeladen werden sollen, bei UDSKDBS-Anwendungen zusätzlich die Bibliothek, aus der die PLITAB-Module nachgeladen werden sollen.

Die Datenmodule brauchen nicht (wie in den UDS/SQL-Vorgängerversionen) in einer einzigen Bibliothek abgelegt sein, sondern können in verschiedenen Bibliotheken gehalten werden, beispielsweise pro Datenbank in einer eigenen.

Die Zuweisung der Bibliothek(en) erfolgt über Linknamen-Zuweisungen mit dem ADD-FILE-LINK-Kommando.

SSITAB-Modul:

1. Als Erstes wird die Bibliothek durchsucht, die mit dem Linknamen $UDSSSI zugewiesen wurde.
2. Wenn die SSITAB-Module nicht nur in einer Bibliothek gehalten werden, z.B. pro Datenbank in einer eigenen, können die weiteren Bibliotheken mit den Linknamen BLSLIB00 bis BLSLIB99 zugewiesen werden.

Die Zuweisung mit dem Linknamen $UDSSSI und ggf. zusätzlich mit BLSLIB*nn* ist das von uns empfohlene Standardverfahren. Für dieses Verfahren wird in den Fehlermeldungen des Produkts die Abkürzung „$UL“ verwendet.

3. Wenn der Linkname $UDSSSI nicht verwendet wird oder der Nachladevorgang nicht erfolgreich war, wird aus Kompatibilitätsgründen eine in der Ablaufkennung vorhandene Bibliothek UDS.MODLIB bzw. eine mit dem SET-TASKLIB-Kommando zugewiesene Bibliothek durchsucht. Für dieses Verfahren wird in den Fehlermeldungen des Produkts die Abkürzung „TSK“ verwendet.

PLITAB-Modul:

Als Erstes wird die Bibliothek durchsucht, die mit dem Linknamen $UDSPLEX zugewiesen wurde. Die weitere Suche in den mit dem Linknamen BLSLIB*nn* zugewiesenen Bibliotheken bzw. in einer Bibliothek UDS.MODLIB erfolgt analog zu den SSITAB-Modulen.

Nachladen des konfigurationsspezifischen Tabellenmoduls UDSTRTAB

Wenn eine Umsetztabelle (UDSTRTAB) für eine benutzerspezifische Sortierung von Character-Feldern vom DBH genutzt werden soll, muss die Bibliothek bekannt sein, aus der das UDSTRTAB-Modul nachgeladen werden soll. Als Erstes wird eine Bibliothek in der Konfigurationskennung durchsucht, die mit dem Linknamen $UDSKONF zugewiesen wurde. Da diese Tabelle vom DBH verwendet wird, muss die Zuweisung einer Bibliothek nur beim Start von linked-in-Anwendungen erfolgen.

Für weitere Einzelheiten siehe Abschnitt „Umsetztabelle für eine anwendungsspezifische Sortierung“ auf Seite 126.

6.4.2 UDS/SQL-TIAM-Anwendungen binden

Dieser Abschnitt beschreibt das Binden eines UDS/SQL-TIAM-Anwenderprogramms. Informationen zum Binden einer UDS/SQL-openUTM-Anwendung finden Sie auf Seite 109.

Damit ein UDS/SQL-TIAM-Anwenderprogramm ausgeführt werden kann, muss die Verbindung zum UDS/SQL-DBH hergestellt werden. Zu diesem Zweck stellt UDS/SQL Verbindungsmodule bereit, die in die UDS/SQL-Anwendung eingebunden werden.

Versionsunabhängige Verbindungsmodule in die UDS/SQL-Anwendung einbinden

UDS/SQL stellt die versionsunabhängigen Verbindungsmodule UDSLNKI, UDSLNKL und UDSLNKA bereit, die in die UDS/SQL-TIAM-Anwendung eingebunden werden. Diese versionsunabhängigen Verbindungsmodule enthalten die Ansprungstellen für die Verarbeitung der DML-Anweisungen und laden die benötigten versionsabhängigen Verbindungsmodule (UDSBCCON bzw. LCCONCT) nach.

Tabelle 12 zeigt abhängig vom verwendeten DBH, welche versionsunabhängigen Verbindungmodule eingebunden werden und welche versionsabhängigen Verbindungsmodule daraus dynamisch nachgeladen werden. Wenn das Modul UDSLNKA eingebunden wird, muss vor Aufruf des Anwenderprogramms mit dem Kommando MODIFY-JOB-SWITCHES angegeben werden, welche DBH-Variante genutzt werden soll.

verwendeter DBH	eingebundenes Modul	nachgeladenes Modul
independent DBH	UDSLNKI	UDSBCCON
	UDSLNKA Auftragsschalter 28 = ON	
linked-in DBH	UDSLNKL	LCCONCT
	UDSLNKA Auftragsschalter 28 = OFF	

Tabelle 12: versionsunabhängige und versionsabhängige Verbindungsmodule

Wenn nur die versionsunabhängigen Verbindungsmodule in die UDS/SQL-Anwendung eingebunden werden, kann die UDS/SQL-Anwendung bei sonst gleicher Systemumgebung unverändert mit einer neuen UDS/SQL-Version oder mit einer UDS/SQL-Korrekturversion eingesetzt werden. Somit können Sie die funktionalen Erweiterungen einer neuen UDS/SQL-Version nutzen, ohne das Anwenderprogramm sofort neu binden zu müssen.

Technisch gesehen können statt der versionsunabhängigen die versionsabhängigen Verbindungsmodule statisch in das Programm eingebunden werden (siehe Tabelle 12: nachgeladenes Modul). Dies ist jedoch nicht die von uns empfohlene Methode, da bei jedem UDS/SQL-Versionswechsel neu gebunden werden muss.

Für besondere Einsatzfälle sind in den Verbindungsmodulen Weak Externs enthalten. Die Nichtbefriedigung dieser Adressverweise verhindert nicht den Start der Anwendung:

– Die Befriedigung von DSCEXT wird bei der Fehlerbehandlung von CALL-DML-Aufrufen benötigt (siehe Abschnitt „Fehlerbehandlungsroutine DSCEXT der CALL-DML“ auf Seite 124).

– Einen Hinweis auf Nichtbefriedigung von UDS@UNR# können Sie ignorieren.

Beispiel

Das folgende Beispiel zeigt das Binden einer UDS/SQL-Anwendung (COBOL-DML) mit dem independent DBH.

```
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=modul
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=bibliothek-1
  ,ELEMENT=element)
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=udssyslnklib
  ,ELEMENT=UDSLNKI)
//RESOLVE-BY-AUTOLINK LIBRARY=crtesyslnk
//SAVE-LLM MODULE-CONTAINER=*LIB(LIBRARY=bibliothek-2,ELEMENT=modul)
//END
```

Behandlung von Namenskonflikten

Wenn beim Binden von Anwendungen, die ein versionsunabhängiges Verbindungsmodul UDSLNK*x* nutzen, die Entries des Moduls UDSLNK*x* sichtbar bleiben, kann es beim weiteren Nachladen der versionsabhängigen UDS/SQL-Module zu Namenskonflikten kommen. Zur Behebung dieser Namenskonflikte genügt es in der Regel, beim Start der Anwendungen das Kommando START-EXECUTABLE-PROGRAM mit der Angabe DBL-PAR=(ERROR-PROC(NAME-COLLISION=*STD)) zu verwenden (siehe auch Kommando START-EXECUTABLE-PROGRAM im Abschnitt „COBOL-Programm starten“ auf Seite 103ff). Wenn aus anwendungsspezifischer Sicht diese „generelle“ Angabe beim Start der Anwendung nicht möglich ist, können die UDS/SQL-spezifischen Namenskonflikte gezielt beim Binden mit dem BINDER behandelt werden:

1. Die Entries des versionsunabhängigen Moduls UDSLNK*x* ($UNIASE, $UNIBASE, DML, DMLTRACE, KDBSFITA, KKDS, KLDS, LINDA, SQLUDS) werden beim Binden mit dem BINDER maskiert. Diese Variante eignet sich für Anwendungen, bei denen alle UDS/SQL-Aufrufe aus einer Ladeeinheit erfolgen.

Beispiel

```
/START-BINDER
...
//MODIFY-SYMBOL-VISIBILITY SYMBOL-NAME=($UNIASE,$UNIBASE,DML -
//,DMLTRACE,KDBSFITA,KKDS,KLDS,LINDA,SQLUDS),VISIBLE=*NO()
...
//END
```

2. Die Entries des versionsunabhängigen Moduls UDSLNK*x* ($UNIASE, $UNIBASE, DML, DMLTRACE, KDBSFITA, KKDS, KLDS, LINDA, SQLUDS) werden beim Binden mit dem BINDER anwendungsspezifisch umbenannt. Diese Variante eignet sich für Anwendungen, bei denen die UDS/SQL-Aufrufe aus mehreren Ladeeinheiten erfolgen. In jeder Ladeeinheit werden nur die dort jeweils bereitgestellten bzw. referenzierten Entries umbenannt, die restlichen Entries werden analog zu 1. maskiert.

   Beispiel

```
/START-BINDER
...
//RENAME-SYMBOLS SYMBOL-NAME=SQLUDS,NEW-NAME=UDSSQL -
//,SYMBOL-OCCURRENCE=*PAR(OCCURRENCE-NUMBER=*ALL)
...
//END
```

6.4.3 COBOL-Programm starten

Sie können folgende DBH-Varianten benutzen:

bei	**independent DBH**	**linked-in DBH**
Mono-DB	X	X
Multi-DB	X	X
openUTM	X	-

Tabelle 13: DBH-Varianten

1. Starten eines Anwenderprogramms mit independent DBH

```
[/MODIFY-JOB-SWITCHES ON=28] ————————————————————————————————————  (1)

 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version ———————  (2)
[/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D,VERSION=version]

                                                          ⎧dbname          ⎫
 /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=             ⎨                ⎬ —  (3)
                                                          ⎩konfigurationsname⎭

[/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=SSITAB-bibliothek-1] ——  (4)
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=SSITAB-bibliothek-nn]

[/ADD-FILE-LINK LINK-NAME=$UDSPLEX,FILE-NAME=PLITAB-bibliothek-1] —  (5)
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=PLITAB-bibliothek-nn]

/START-EXECUTABLE-PROGRAM FROM-FILE=(LIBRARY=bibliothek-2, ————————  (6)
 ELEMENT=modul),DBL-PAR=(ERROR-PROC(NAME-COLLISION=*STD))

 [Anwenderprogramm-Parameter]
```

(1) Wenn in das Anwenderprogramm das Verbindungsmodul UDSLNKA eingebunden wurde, muss der Auftragsschalter 28 auf ON gesetzt werden, damit der independent DBH verwendet wird (das versionsabhängige Modul UDSBCCON wird nachgeladen).

(2) Es wird empfohlen, generell mit dem Kommando SELECT-PRODUCT-VERSION anzugeben, welche UDS/SQL-Version genutzt werden soll, da mit IMON im Software Configuration Inventory (SCI) mehrere UDS/SQL-Versionen parallel installiert und mehrere Versionen des UDS/SQL-Subsystems vorgeladen sein können.

Bei Einsatz von UDS-D sollte mit einem zusätzlichen SELECT-PRODUCT-VERSION-Kommando die zur UDS/SQL-Version korrespondierende UDS-D-Version angegeben werden, wenn UDS-D als Subsystem genutzt wird.

Für weitere Einzelheiten siehe Abschnitt „Nachladen der UDS/SQL-Produktmodule“ auf Seite 95.

(3) Für die Ausführung eines UDS/SQL-Anwenderprogramms müssen Sie der betreffenden Anwendung die UDS/SQL-Konfiguration bekannt geben: Mit dem Kommando SET-FILE-LINK weisen Sie die Konfigurationsdatei (FILE-NAME=*konfigurationsname*) über den Linknamen DATABASE zu. Im Mono-DB-Betrieb kann der Konfigurationsname auch mit dem Namen der verwendeten Datenbank übereinstimmen (FILE-NAME=*dbname*).
Im Fall FILE-NAME=*dbname* ist für die Datenbank die Benutzungsart SHARED-RETRIEVAL nur möglich, wenn Sie den Ladeparameter DBNAME angeben (siehe Handbuch „Datenbankbetrieb“).

(4), (5)
Im Falle von CALL-DML-Anwenderprogrammen muss mit dem ADD-FILE-LINK-Kommando die Bibliothek zugewiesen werden, aus der zum Ablaufzeitpunkt die SSITAB-Module nachgeladen werden sollen, bei KDBS-Anwendungen zusätzlich die Bibliothek, aus der die PLITAB-Module nachgeladen werden sollen.
Für weitere Einzelheiten siehe Abschnitt „Nachladen der anwendungsspezifischen Datenmodule SSITAB und PLITAB“ auf Seite 98.

(6) Das Anwenderprogramm wird mit dem Kommando START-EXECUTABLE-PROGRAM gestartet.

Der im RUN-MODE=*ADVANCED voreingestellte Wert NAME-COLLISION=*STD sollte nicht verändert werden, da es sonst bei Anwendungen, die mit dem BINDER gebunden wurden, zu Namenskonflikten bei nachzuladenden Entries kommen kann (siehe auch „Behandlung von Namenskonflikten“ auf Seite 101).

2. Starten eines Anwenderprogramms mit linked-in DBH

Kommandos, die die Steuerung des DBH selbst betreffen, sind im Folgenden nicht oder nur knapp beschrieben. Die ausführliche Beschreibung dieser Kommandos finden Sie im Handbuch „Datenbankbetrieb“, Abschnitt „DBH-Startkommandos“.

```
[/MODIFY-JOB-SWITCHES OFF=28] ————————————————————————————  (1)

 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version ———  (2)

 /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME={dbname                } —  (3)
                                             {konfigurationsname}

[/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=SSITAB-bibliothek-1] ——  (4)
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=SSITAB-bibliothek-nn]

[/ADD-FILE-LINK LINK-NAME=$UDSPLEX,FILE-NAME=PLITAB-bibliothek-1] —  (5)
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=PLITAB-bibliothek-nn]

[/ADD-FILE-LINK LINK-NAME=$UDSKONF,FILE-NAME=UDSTRTAB-bibliothek] —  (6)

 /CREATE-FILE FILE-NAME=konfname.DBSTAT,SUPPRESS-ERR=*FILE-EXISTING   (7)
 /CREATE-FILE FILE-NAME=konfname.DBSTAT.SAVE,SUPPRESS-ERR=*FILE-EXISTING

[/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=DBH-parameterdatei]———  (8)

[weitere DBH-spezifische Kommandos] ———————————————————————  (9)

/START-EXECUTABLE-PROGRAM FROM-FILE=(LIBRARY=library-2, ———————— (10)
 ELEMENT=modul),DBL-PAR=(ERROR-PROC(NAME-COLLISION=*STD))
[Anwenderprogramm-Parameter]
```

(1) Wenn in das Anwenderprogramm das Verbindungsmodul UDSLNKA eingebunden wurde, muss der Auftragsschalter 28 auf OFF gesetzt werden, damit der linked-in DBH verwendet wird (das versionsabhängige Modul LCCONCT wird nachgeladen).

(2) Es wird empfohlen, generell mit dem Kommando SELECT-PRODUCT-VERSION anzugeben, welche UDS/SQL-Version genutzt werden soll, da mit IMON im Software Configuration Inventory (SCI) mehrere UDS/SQL-Versionen parallel installiert und mehrere Versionen des UDS/SQL-Subsystems vorgeladen sein können.

Für weitere Einzelheiten siehe Abschnitt „Nachladen der UDS/SQL-Produktmodule“ auf Seite 95.

(3) Analog zum Starten eines Anwenderprogramms mit independent DBH muss die UDS/SQL-Konfiguration bekannt gegeben werden. Siehe unter (3) auf Seite 104.

(4), (5)
Im Falle von CALL-DML-Anwenderprogrammen muss mit dem ADD-FILE-LINK-Kommando die Bibliothek zugewiesen werden, aus der zum Ablaufzeitpunkt die SSITAB-Module nachgeladen werden sollen, bei KDBS-Anwendungen zusätzlich die Bibliothek, aus der die PLITAB-Module nachgeladen werden sollen.
Für weitere Einzelheiten siehe Abschnitt „Nachladen der anwendungsspezifischen Datenmodule SSITAB und PLITAB“ auf Seite 98.

(6) Wenn eine Umsetztabelle (UDSTRTAB) für eine benutzerspezifische Sortierung von Character-Feldern genutzt wird, muss die Bibliothek bekannt sein, aus der das UDSTRTAB-Modul nachgeladen werden soll. Als Erstes wird eine Bibliothek in der Konfigurationskennung durchsucht, die mit dem Linknamen $UDSKONF zugewiesen wurde. Für weitere Einzelheiten siehe Abschnitt „Umsetztabelle für eine anwendungsspezifische Sortierung“ auf Seite 126.

(7) Die DB-Status-Dateien müssen, falls noch nicht vorhanden, eingerichtet werden. Die DB-Status-Dateien werden in 4-Kbyte-Seitenformat eingerichtet. Sie können ihre Lage auf getrennten Datenträgern bestimmen.

Wenn die DB-Status-Dateien auf privater Platte liegen sollen, so müssen Sie diese mit folgendem Kommando einrichten:

```
/CREATE-FILE FILE-NAME=konfname.DBSTAT[.SAVE]
 ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn
 ,DEVICE-TYPE=gerät[,SPACE=...])
 ,SUPPRESS-ERRORS=*FILE-EXISTING
```

(8) Mit dem Linknamen PPFILE wird eine Datei zugewiesen, aus der der linked-in DBH die Ladeparameter einlesen soll. Die Ladeparameter können auch in einem PLAM-Bibliothekselement oder in einer Prozedurdatei stehen. Die hierfür benötigten Zuweisungskommandos sind dem Handbuch „Datenbankbetrieb“, Abschnitt „DBH-Startkommandos“ zu entnehmen.

(9) Weitere, wahlweise anzugebende Kommandos zur Steuerung des linked-in DBH sind ausführlich im Handbuch „Datenbankbetrieb“, Abschnitt „DBH-Startkommandos“ beschrieben.

(10) Das Anwenderprogramm wird mit dem Kommando START-EXECUTABLE-PROGRAM gestartet.

Der voreingestellte Wert NAME-COLLISION=*STD sollte nicht verändert werden, da es sonst bei Anwendungen, die mit dem BINDER gebunden wurden, zu Namenskonflikten bei nachzuladenden Entries kommen kann (siehe auch „Behandlung von Namenskonflikten“ auf Seite 101).

6.5 Zusammenarbeit bei einer UDS/SQL-openUTM-Anwendung

UDS/SQL-openUTM-Anwendung generieren

Das Generieren einer UDS/SQL-openUTM-Anwendung umfasst die gleichen Schritte wie das Generieren einer openUTM-Anwendung.

Informationen zum Generieren einer UDS/SQL-openUTM-Anwendung finden Sie im openUTM-Handbuch „Anwendungen generieren“.

Die einzelnen Schritte sind:

– Definieren der Anwendungskonfiguration und Erstellen des KDCROOT-Moduls mit dem Dienstprogramm KDCDEF
– Übersetzen der Main-Routine KDCROOT
– Übersetzen der Anwendungsteilprogramme
– Binden der Module zum Anwendungsprogramm

KDCDEF-Anweisung DATABASE

Wesentlich für die Zusammenarbeit von openUTM und UDS/SQL ist, dass mit der KDCDEF-Anweisung DATABASE der Datenbankanschluss in der Main-Routine KDCROOT generiert wird:

```
DATABASE [ENTRY=entry][,LIB=LOGICAL-ID(SYSLNK)][,TYPE=type]
```

ENTRY=*entry*
: bezeichnet den ENTRY-Namen der Datenbank;
Sie geben den ENTRY-Namen für die verwendete UDS/SQL-Schnittstelle an:

  – $UNIBASE für COBOL-DML (Standardwert)
  – DML für CALL-DML
  – KKDS für KDBS
  – SQLUDS für SQL

LIB=LOGICAL-ID(SYSLNK)
: Das Verbindungsmodul wird aus der als IMON-Installationspfad für SYSLNK eingetragenen Bibliothek nachgeladen. Dabei wird eine über SELECT-PRODUCT-VERSION zugewiesene UDS/SQL-Version oder standardmäßig die höchste UDS/SQL-Version genutzt.
Weitere Möglichkeiten zur Versorgung des Parameters LIB finden Sie im openUTM-Handbuch „Anwendungen generieren“.

TYPE=*type*
bezeichnet den Typ des Datenbanksystems.
Sie geben hier UDS an. Sofern Sie in einer openUTM-Transaktion sowohl die SQL- als auch eine CODASYL-Schnittstelle nutzen, ist auch die Angabe DB zulässig (siehe „Zwei UDS/SQL-Konfigurationen in einer UDS/SQL-openUTM-Anwendung“ auf Seite 113).

Übersetzen der Main-Routine KDCROOT

Bei der Übersetzung von KDCROOT müssen UDS/SQL- und openUTM-Makrobibliotheken zugewiesen werden.

Beispiel für eine Übersetzung mit dem ASSEMBH

```
/START-ASSEMBH
//COMPILE SOURCE=rootname,MACRO-LIB=(utmmacrolib -
// ,udssyslib),MODULE-LIB=rootbibliothek)
//END
```

UDS/SQL-openUTM-Anwendung binden

Beim Binden einer UDS/SQL-openUTM-Anwendung können Sie zwischen drei unterschiedlichen Varianten auswählen:

- Verbindungsmodul UDSCON wird dynamisch durch openUTM nachgeladen
- Verbindungsmodul UDSCON wird dynamisch durch UDS/SQL nachgeladen
- Verbindungsmodul UDSCON wird fest eingebunden

Diese drei Varianten werden im Folgenden erläutert:

Verbindungsmodul UDSCON wird dynamisch durch openUTM nachgeladen

Dabei kann der Hinweis des BINDER `UNRESOLVED EXTRNS: UDSCON` ignoriert werden. openUTM lädt das UDS/SQL-Verbindungsmodul UDSCON dynamisch nach. Der im Subsystem von UDS/SQL enthaltene Verbindungsmodul kann ressourcenschonend genutzt werden. Bei der Nutzung einer neuen Korrekturversion muss die UDS/SQL-openUTM-Anwendung nicht neu gebunden werden.

Beispiel

```
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=modul
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=rootbibliothek -
// ,ELEMENT=KDCROOT-modul)
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=bibliothek -
// ,ELEMENT=(teilprog-1,teilprog-2,...teilprog-n))
//RESOLVE-BY-AUTOLINK LIBRARY=bibliothek
//RESOLVE-BY-AUTOLINK LIBRARY=(utmsyslnklib,crtesyslnklib[.PARTIAL-BIND] -
// ,utmsyslnksplrtslib)
//SAVE-LLM MODULE-CONTAINER=*LIB(LIBRARY=bibliothek -
// ,ELEMENT=element,OVERWRITE=YES)
//END
```

Verbindungsmodul UDSCON wird dynamisch durch UDS/SQL nachgeladen

In die UDS/SQL-openUTM-Anwendung wird das versionsunabhängige Modul UDSCNUV eingebunden. Die im Modul KDCROOT enthaltene Referenz auf UDSCON muss auf UDSCNUV umbenannt werden. UDSCNUV lädt das UDS/SQL-Verbindungsmodul dynamisch nach (siehe auch Abschnitt „Nachladen der UDS/SQL-Produktmodule“ auf Seite 95). Das im Subsystem von UDS/SQL enthaltene Verbindungsmodul UDSCON kann ressourcenschonend genutzt werden. Bei der Nutzung einer neuen Korrekturversion muss die UDS/SQL-openUTM-Anwendung nicht neu gebunden werden. Der Modul, in den UDSCNUV eingebunden wird, kann nicht shared genutzt werden.

Beispiel

```
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=modul
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=rootbibliothek -
// ,ELEMENT=KDCROOT-modul)
//RENAME-SYMBOLS SYM-NAM=UDSCON,SYM-TYP=ALL,NEW-NAME=UDSCNUV
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=bibliothek -
// ,ELEMENT=(teilprog-1,teilprog-2,...teilprog-n))
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=udssyslnklib -
// ,ELEMENT=UDSCNUV)
//RESOLVE-BY-AUTOLINK LIBRARY=bibliothek
//RESOLVE-BY-AUTOLINK LIBRARY=(utmsyslnklib,crtesyslnklib[.PARTIAL-BIND] -
// ,utmsyslnksplrtslib)
//SAVE-LLM MODULE-CONTAINER=*LIB(LIBRARY=bibliothek -
// ,ELEMENT=element,OVERWRITE=YES)
//END
```

Verbindungsmodul UDSCON wird fest eingebunden

Bei jedem UDS/SQL-Versionswechsel, auch bei Korrekturversionen, muss neu gebunden werden. Beim Start der UDS/SQL-openUTM-Anwendung ist kein Laden des Verbindungsmoduls UDSCON notwendig. Der im Subsystem von UDS/SQL enthaltene Verbindungsmodul UDSCON kann nicht genutzt werden.

Beispiel

```
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=modul
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=rootbibliothek -
// ,ELEMENT=KDCROOT-modul)
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=bibliothek -
// ,ELEMENT=(teilprog-1,teilprog-2,...teilprog-n))
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=udssyslnklib -
// ,ELEMENT=UDSCON)
//RESOLVE-BY-AUTOLINK LIBRARY=bibliothek
//RESOLVE-BY-AUTOLINK LIBRARY=(utmsyslnklib,crtesyslnklib[.PARTIAL-BIND] -
// ,utmsyslnksplrtslib)
//SAVE-LLM MODULE-CONTAINER=*LIB(LIBRARY=bibliothek -
// ,ELEMENT=element,OVERWRITE=YES)
//END
```

Weitere Informationen zum Binden einer UDS/SQL-openUTM-Anwendung finden Sie im openUTM-Handbuch „Anwendungen generieren“, zum Binden des CRTE im Handbuch „CRTE (BS2000)“.

UDS/SQL-openUTM-Anwendung starten

```
      .
      .
[01  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version]
[02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D,VERSION=version]
      .
      .
[03  /ADD-FILE-LINK LINK-NAME=$UDSLIB,FILE-NAME=udslib]
[04  /ADD-FILE-LINK LINK-NAME=$UDSDLIB,FILE-NAME=udsdlib]
[05  /ADD-FILE-LINK LINK-NAME=$UDSKLIB,FILE-NAME=udskdbslib]
      .
      .
 06  .UDS DATABASE=konfname1
[07  .DB␣␣DATABASE=konfname2]
```

01), 02)
: Da verschiedene Versionen der Subsysteme UDS-SQL und/oder UDS-D parallel vorgeladen sein können, sollte mit SELECT-PRODUCT-VERSION immer die gewünschte Produktversion bekannt gegeben werden. Dabei ist es möglich, über Jobvariablen die jeweils gewünschte Subsystemversion beim DBH-Start zu laden (siehe Handbuch „Datenbankbetrieb“, Abschnitt „Mehrere UDS/SQL-Versionen parallel nutzen“). Ohne Angabe von SELECT-PRODUCT-VERSION wird das Subsystem mit der höchsten Version verwendet. Wenn im Falle von UDS/SQL keine entsprechende Subsystemversion vorgeladen ist und weder UDSCON noch UDSCNUV fest eingebunden ist, wird das Verbindungsmodul UDSCON aus der in der KDCDEF-Anweisung DATABASE festgelegten UDS/SQL-Bibliothek (LIB=LOGICAL-ID(SYSLNK)) nachgeladen.

03), 04), 05)
: Diese Zuweisungen von Bibliotheken einer Privatinstallation müssen nur erfolgen, wenn die verwendete UDS/SQL-Version nicht im SCI verfügbar ist und CALL-DML ($UDSLIB) mit UDS-D ($UDSDLIB) oder UDSKDBS ($UDSKLIB) genutzt wird.

06), 07)
: Start-Parameter, die von UTM an UDS gegeben werden.

  Konfigurationsnamen (*konfname1*, *konfname2*) dürfen in den Startparametern nur ohne Kennung angegeben werden.

  Den Startparameter .DB müssen Sie nur angeben, wenn Sie eine Database-Anweisung mit TYPE=DB generiert haben. Die zwei Leerzeichen nach .DB sind zwingend.

Weitere Informationen zum Starten einer UDS/SQL-openUTM-Anwendung finden Sie im openUTM-Handbuch „Anwendungen generieren“.

Beispiel

```
/SET-LOGON-PARAMETERS
/ADD-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=SYSLOG
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=02.8A00
/ASSIGN-SYSDTA TO=*SYSCMD
/START-EXE-PROG FROM-FILE=*LIB(LIBRARY=LIB.TEST, ELEMENT=AUFABW) -
/ ,DBL-PAR=(LOADING=(LOAD-INFORMATION=REFERENCES)-
/ ,ERROR-PROC=UNRESOLVED-EXTRNS=DELAY))
.UTM START FILEBASE = KDCFILE.AUFABW
.UTM START STARTNAME=MANUAL.STARTEN
.UTM END
.UDS DATABASE=DBHSQL
.FHS MAPLIB=LIB.TEST
END
/EXIT-JOB
```

Zwei UDS/SQL-Konfigurationen in einer UDS/SQL-openUTM-Anwendung

Es ist möglich, in einer UDS/SQL-openUTM-Anwendung innerhalb einer openUTM-Transaktion sowohl SQL-Aufträge als auch CODASYL-/CALL-DML-Aufträge mit UDS/SQL zu bearbeiten. UDS/SQL wird dabei als Typ UDS und als Typ DB (TYPE = DB) in der openUTM-Anwendung generiert.

Voraussetzungen

– Beim Assemblieren der Main-Routine KDCROOT muss die UDS/SQL-Makrobibliothek (SYSLIB.UDS-SQL.*nnn*) zugewiesen werden, die neben dem Makro KDCDBU auch den Makro KDCDBD enthält.

– In einer Transaktion dürfen SQL-Aufträge und CODASYL-/CALL-DML-Aufträge nur gemischt werden, wenn diese vollständig an verschiedene UDS/SQL-Konfigurationen übergeben werden.

Der ENTRY-Name SQLUDS darf nur bei der Definition mit Typ UDS angegeben werden.

Fehlercodes

Bei einer UDS/SQL-openUTM-Anwendung werden in einem speziellen Bereich (DB-DIAGAREA bzw. DB-Record) Ablaufinformationen für Diagnosezwecke hinterlegt (siehe openUTM-Handbuch „Meldungen, Test und Diagnose (BS2000)“). Dabei handelt es sich um versionsabhängige Diagnoseinformationen. Diese Codes werden von openUTM ausgegeben.

Siehe auch Abschnitt „Returncodes bei UDS/SQL-openUTM“ auf Seite 394 und Abschnitt „Zusätzliche Diagnoseinformation bei openUTM“ auf Seite 398.

6.6 Fehlerbehandlung

6.6.1 Unterbrechungsbehandlung bei UDS/SQL-TIAM-Anwendungen

Die in diesem Abschnitt beschriebene Unterbrechungsbehandlung betrifft nur UDS/SQL-TIAM-Anwendungen; zur Fehlerbehandlung bei UDS/SQL-openUTM-Anwendungen siehe Seite 113. Die von UDS/SQL zur Verfügung gestellte Unterbrechungsbehandlung mit Hilfe der Routine SCSXUSER verhindert bei definierten STXIT-Ereignisklassen den Programmabbruch durch das Betriebssystem. Außerdem kann der Anwender zusätzlich eigene Unterbrechungsroutinen festlegen.

Unterbrechungsbehandlung durch UDS/SQL mit SCSXUSER (UDS-STXIT)

Die UDS-STXIT-Routine SCSXUSER behandelt die folgenden Unterbrechungsursachen (siehe Handbuch „Makroaufrufe an den Ablaufteil“):

- Programmfehler (Programmprüfung)
- Unterbrechung durch Intervallzeitgeber
- Ende der Programmlaufzeit
- Nicht behebbare Programmfehler
- Mitteilung an das Programm (INFORM-PROGRAM)
- BREAK/ESCAPE
- Abnormal End (ABEND)
- Normale Programmbeendigung

Bei bestimmten STXIT-Unterbrechungsereignissen werden für das gesamte Datenbanksystem wichtige Maßnahmen durchgeführt, z.B. Rollback der betroffenen Transaktion.

Die UDS-STXIT-Routine SCSXUSER nimmt automatisch an der STXIT-Parallelität teil.

Außerdem können Sie eine STXIT-Routine für eine zusätzliche Unterbrechungsbehandlung definieren. Diese STXIT-Routine sollte ebenfalls die STXIT-Parallelität nutzen und sich mit EXIT CONTINU=YES beenden, damit nicht die Ausführung der SCSXUSER-Routine verhindert wird.
Bei der STXIT-Ereignisklasse „Mitteilungen an das Programm“ mit dem INFORM-PROGRAM-Kommando prüft SCSXUSER (UDS-STXIT), ob die Mitteilung für UDS/SQL bestimmt ist. Ist das der Fall, wird die SCSXUSER mit EXIT CONTINU=NO beendet. Die nachfolgenden STXIT-Routinen für dieses Ereignis kommen nicht zum Ablauf.
Bei der Unterbrechung BREAK/ESCAPE läuft das Programm auf einen definierten Unterbrechungspunkt. Die Register zeigen dann den Stand, der zum Zeitpunkt der Unterbrechung vorliegt. Das Programm kann mit TRACE oder RESUME fortgesetzt werden.

Die Benutzung von UDS-STXIT durch SCSXUSER bringt folgende Vorteile:

- Aus Sicht des Gesamtsystems:
  Offene Transaktionen, die durch illegales Beenden oder Fehler des Anwenderprogramms oder durch Fehlen einer abschließenden FINISH-Anweisung nicht beendet wurden, werden zurückgesetzt. Die durch den Abbruch belegten Transaktionskanäle (siehe Handbuch „Datenbankbetrieb“, DBH-Ladeparameter TRANSACTION) werden dadurch frei (jede nicht abgeschlossene Transaktion reduziert sonst die Zahl parallel durchführbarer Transaktionen um 1).
- Aus der Sicht des betroffenen Anwenderprogramms:
  Das Zurücksetzen mit SCSXUSER erleichtert es dem Anwenderprogramm, eine Transaktion mit Hilfe einer eigenen STXIT-Routine zu überspringen (z.B. bei Datenfehlern); wenn kein Arbeiten mit der Datenbank mehr möglich ist, kann das Anwenderprogramm datenbankunabhängige Arbeiten weiterhin ausführen.

Wenn SCSXUSER nicht nachgeladen werden kann, gibt das Verbindungsmodul eine entsprechende Warnung aus, fährt aber mit der Verarbeitung fort. In Privatinstallationen (vgl. Seite 97) ist es möglich, UDS/SQL-TIAM-Anwendungen ohne die UDS/SQL-Unterbrechungsbehandlung zu betreiben. Anwendungen, die über openUTM mit UDS/SQL verbunden sind, arbeiten nicht mit SCSXUSER. Für diese Anwendungen übernimmt openUTM die Unterbrechungsbehandlung.

Unterbrechungsbehandlung durch den Anwender

Zusätzlich zur Unterbrechungsbehandlung von UDS/SQL durch SCSXUSER können Sie mit dem Makro STXIT und Operand STXDNEW eigene STXIT-Routinen für die gewünschten Ereignisklassen definieren.

Diese STXIT-Routinen beenden Sie mit dem Makro EXIT und den Operanden CONTINU=YES,TERM=NO.

Ausnahmen

– STXIT-Routinen für die Ereignisklassen „Programmfehler“ oder „Programmüberprüfung“ sollen bei STXIT-Parallelität mit dem Makro EXIT,CONTINU=YES, TERM=(STEP) beendet werden.

– Eine STXIT-Routine für die Ereignisklasse „Mitteilung an das Programm“ sollte mit dem Makro EXIT und Operand CONTINU=NO,TERM=NO beendet werden, wenn eine für diese Routine bestimmte Mitteilung bearbeitet wurde.

VORSICHT!
In STXIT-Routinen ist es nicht erlaubt, DML-Anweisungen an die DB zu senden, da die UDS-STXIT SCSXUSER in jedem Fall versucht, die Transaktion zu beenden. Kommen STXIT-Routinen von Anwenderprogrammen und anderen Systemen zum

Ablauf, die nicht auf STXIT-Parallelität angepasst sind, ist es unsicher, ob alle STXIT-Routinen für das eingetretene Ereignis gestartet werden. Es können dann eventuell auch Transaktionskanäle gesperrt bleiben und für den weiteren Sessionabschnitt nicht mehr verfügbar gemacht werden.
Meldungen und Speicherauszüge können von anderen Systemen kommen, da beim Eintritt eines Unterbrechungsereignisses alle für dieses Ereignis definierten STXIT-Routinen aktiviert werden.

6.6.2 Datenbanksonderzustände

Verwendung der Sonderregister (COBOL-DML) bzw. des Benutzerinformationsbereichs (CALL-DML)

- DATABASE-REALM-NAME
  Hier steht ein Realm-Name, der mit dem Datenbanksonderzustand zusammenhängt, wenn eine DML-Anweisung nicht erfolgreich beendet werden kann.
- DATABASE-SET-NAME
  Hier steht ein Setname, der mit dem Datenbanksonderzustand zusammenhängt, wenn eine DML-Anweisung nicht erfolgreich beendet werden kann.
- DATABASE-RECORD-NAME
  Hier steht ein Satzname, der mit dem Datenbanksonderzustand zusammenhängt, wenn eine DML-Anweisung nicht erfolgreich beendet werden kann.
- DATABASE-STATUS
  Hier steht der Datenbank-Statusanzeiger, der bei der Ausführung jeder DML-Anweisung neu versorgt wird. In den linken zwei Zeichen steht der Anweisungscode, in den rechten drei Zeichen der Statuscode.
  Welcher Statuscode zu welchem Datenbanksonderzustand gehört, zeigt der Abschnitt „Statuscodes“ auf Seite 367.
  Endet die Ausführung irgendeiner DML-Anweisung mit einem Datenbanksonderzustand, so gibt der Anweisungscode an, welche DML-Anweisung den Sonderzustand verursacht hat, und der Statuscode zeigt an, welcher Sonderzustand aufgetreten ist. Wird eine DML-Anweisung erfolgreich durchgeführt, so erhalten sowohl Anweisungscode wie auch Statuscode den Wert Null.
  Eine Ausnahme bilden Sonderzustände, die vom UDS/SQL-Verbindungsmodul erkannt werden. Hier enthält der Anweisungscode immer den Wert Null (’00’).

Die Sonderregister DATABASE-REALM-NAME und DATABASE-RECORD-NAME werden bei FIND- und STORE-Anweisungen auch dann verwendet, wenn kein Datenbanksonderzustand eingetreten ist. Sie enthalten den Namen des betroffenen Realm und des betroffenen Satzes, im DATABASE-STATUS steht dann der Wert Null.

Der Inhalt der Sonderregister DATABASE-REALM-NAME, DATABASE-SET-NAME und DATABASE-RECORD-NAME ist nicht interpretierbar, wenn der DBH keinen Realm-, Satz, oder Setnamen erkennen kann, weil die DML-Anweisung darüber keine Auskunft gibt.

Anweisungscodes

Bei COBOL-DML finden Sie den Anweisungscode im Sonderregister DATABASE-STATUS. Bei CALL-DML wird das Ergebnisfeld des Parameters Benutzerinformation in den ersten zwei Stellen mit dem Anweisungscode belegt.

	Anweisungen der	
Anweisungscode	**COBOL-DML**	**CALL-DML**
01	CONNECT	CONNEC
02	DISCONNECT	DISCON
03	ERASE	ERASEC
04	FIND/FETCH	FIND1/FTCH1, FIND1L/FTCH1L FIND2/FTCH2 FIND3/FTCH3 FIND4/FTCH4 FIND5/FTCH5 FIND6/FTCH6 FIND7A/FTCH7A
05	FINISH	FINISC
06	FREE	FREEC
07	GET	GETC
08	IF	IFC
09	KEEP	KEEPC
10	MODIFY	MODIF1 MODIF2
12	READY	READYC
14	STORE	STORE1, STOR1L STORE2, STOR2L
15	ACCEPT	ACCPTC, ACCPTL
16	SET	–
25	–	LOOKC
00	Bei allen vom Verbindungsmodul vergebenen Sonderzuständen	

Tabelle 14: Zuordnung der Anweisungscodes zu den Funktionen

Die UDS-Online-Utility (siehe Handbuch „Sichern, Informieren und Reorganisieren“) verwendet für ihre spezifischen DMLs den Anweisungscode 13.

Statuscodes

Die folgende Tabelle zeigt Ihnen das Verhalten von UDS/SQL bei den verschiedenen Statuscodes:

	Statuscodes					
	000	**001**	**018, 113, 122, 218**	**≠ 000 bei READY**	**200**	**alle anderen**
DML-Anweisung erfolgreich gewesen	ja	ja	nein	nein	vorläufig	nein
UDS/SQL bricht die Transaktion ab	nein[1)]	nein[1)]	ja	ja	nein	nein[1)]
der Inhalt des Satzbereichs der UWA ist ...	entsprechend der Anweisung	der nächste Satz	undefiniert, weil die Transaktion abgebrochen wurde		unverändert	unverändert

Tabelle 15: Folgen der einzelnen Statuscodes

1 außer Sie führen einen FINISH WITH CANCEL durch

Die Liste der Statuscodes finden Sie im Anhang, siehe Seite 367 für die DML und Seite 380 für die CALL-DML.

Kombinationen von Anweisungscodes und Statuscodes

	Anweisungscode															
Statuscode	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**12**	**13**	**14**	**15**	**16**	**25**
001 010					siehe Tabelle 17 auf Seite 122								X				
011 012 013 018 020		X	X	X			X	X	X	X	X	X	X X X X X	X	X		X
021 022 023 024 027 028 029		X						X			X X	X	X	X X	X		
031 032 033		X X X	X X	X X				X X	X X		X X X			X			
042 043 044		X X	X	X					X		X X	X	X	X X	X		
051		X									X			X			
071 072				X													
081 082 083		X	X X								X X						
091 092 093 099		X X	X X	X X					X		X X	X X X	X X X	X X	X		
101 102 103		X	X	X		X		X	X		X	X	X	X	X X	X	
113	X	X	X	X		X		X	X		X	X	X	X	X		X

Tabelle 16: Kombination von Anweisungscodes und Statuscodes (Teil 1 von 3)

	Anweisungscode																
Statuscode	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**12**	**13**	**14**	**15**	**16**	**25**
122	X	X	X	X	siehe Tabelle 17 auf Seite 122	X	X	X	X	X	X	X	X	X	X		X
123												X	X				
124												X	X				
131	X												X				
132												X	X				
134	X	X	X	X		X	X	X	X	X	X		X	X	X		X
136	X	X	X	X			X	X	X	X	X		X	X	X		X
137	X												X				
141												X	X				
142												X	X				
144	X	X	X	X			X	X	X	X	X		X	X	X		
145												X	X				
146	X	X	X	X			X	X	X	X	X	X	X	X	X		
151	X												X				
152	X												X				
154	X												X				
155	X												X				
161													X				
162													X				
163													X				
164													X				
165													X				
166													X				
167													X				
183																	
184																	
191																	
192																	
193																	
194																	
195																	
197																	
198																	
200	X					X											
201	X					X											
218	X	X	X	X			X	X	X	X	X	X		X	X		X

Tabelle 16: Kombination von Anweisungscodes und Statuscodes (Teil 2 von 3)

	Anweisungscode															
Statuscode	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**12**	**13**	**14**	**15**	**16**	**25**
781					siehe Tabelle 17 auf Seite 122								X				X
782													X				X
783													X				X
784													X				X
785													X				X
786													X				X
789													X				X
802		X									X			X			
804														X			
805		X	X	X						X	X	X	X	X			
888											X			X			
898											X			X			
899								X			X			X			
901		X	X	X				X	X		X	X	X	X			
950												X	X				
954												X	X				

Tabelle 16: Kombination von Anweisungscodes und Statuscodes (Teil 3 von 3)

Statuscodes bei FIND/FETCH

Statuscode	**Formate des FIND/FETCH-Satzauswahlausdrucks**						
04...	**1**	**2**	**3**	**4**	**5**	**6**	**7**
001							X
018	X	X	X	X	X	X	X
020	X	X	X	X	X	X	X
021		X	X	X			
023							X
024	X	X		X			X
027			X				X
028	X						
029				X	X		
031		X	X	X	X	X	X
032					X		
033					X		
042	X		X	X			X
043		X					X
071		X	X	X	X		

Tabelle 17: Kombination von Anweisungscode 04 und Statuscodes (Teil 1 von 2)

Statuscode	Formate des FIND/FETCH-Satzauswahlausdrucks						
04...	**1**	**2**	**3**	**4**	**5**	**6**	**7**
091	X	X	X	X	X	X	X
101				X		X	X
102	X						
103	X	X	X	X	X	X	X
113	X	X	X	X	X	X	X
122	X	X	X	X	X	X	X
134	X	X	X	X	X	X	X
136	X	X	X	X	X	X	X
144	X	X	X	X	X	X	X
146	X	X	X	X	X	X	X
183							X
184			X				X
191							X
192							X
193			X	X			X
194							X
195							X
197			X				
198			X				
218	X	X	X	X	X	X	X
805							X
901	X	X	X	X	X	X	X

Tabelle 17: Kombination von Anweisungscode 04 und Statuscodes (Teil 2 von 2)

6.6.3 Fehlerbehandlungsroutine DSCEXT der CALL-DML

Die Fehlerbehandlungsroutine DSCEXT der CALL-DML betrifft nur UDS/SQL-TIAM-Anwendungen.
Wenn der Parameter Benutzerinformation nicht identifizierbar ist, d.h., das Kennzeichen USINF* bzw. UINF1* fehlt oder steht nicht an der richtigen Stelle im Benutzerinformationsbereich, dann ist keine Kommunikation des Programms mit UDS/SQL möglich und es kann kein Statuscode übergeben werden.
Um diesen Fall zu behandeln, müssen Sie in Ihrem Programm eine Fehlerroutine mit dem ENTRY-Namen DSCEXT definieren.

Wenn Sie den Fehlerausgang DSCEXT nicht definiert haben, wird der Statuscode C91 ausgegeben. Das Programm kann keine DML-Funktionen ausführen.

Wenn während des Programmlaufs der Parameter Benutzerinformation nicht erkannt wird, erscheint die Meldung

```
UDS0283 UDS USER ERROR: USERINF PARAM WRONG OR MISSING
```

und der Fehlerausgang DSCEXT wird angesprungen.

Ist weder der Benutzerinformationsbereich noch der Fehlerausgang DSCEXT vorhanden, erscheint zusätzlich die Meldung

```
UDS0284 UDS USER ERROR: NO DSCEXT-ROUTINE DEFINED
```

und der Programmlauf wird abgebrochen.

CALL-DML verwendet also in diesen Fällen die Rücksprungadresse im Register 14 nicht (siehe Kapitel „Nachschlageteil der CALL-DML“ auf Seite 197). Der CALL-DML-Aufruf wird nicht ausgeführt.

In openUTM-Anwenderprogrammen ist der Fehlerausgang DSCEXT nicht erforderlich und wird nicht benutzt.

Wenn der Fehlerausgang DSCEXT angesprungen wird, ist es in der Regel nicht sinnvoll, die Bearbeitung fortzusetzen, da ein Programmierfehler vorliegt, z.B.:

– falsche Definition des Benutzerinformationsbereichs
– falsche Versorgung der Parameteradressen oder der Übergabe in Register 1
– unbeabsichtigtes Überschreiben des Benutzerinformationsbereichs während des Programmlaufs

Das Ausgeben einer Meldung und das Absetzen eines FINISC bei offener Transaktion ist nicht notwendig, da UDS/SQL selbst eine entsprechende Meldung ausgibt und einen FINISH WITH CANCEL generiert, wenn das Programm mit offener Transaktion beendet wird.

Sie können die DSCEXT-Routine in einem eigenen Modul beschreiben, wenn sie in mehreren CALL-DML-Programmen verwendet werden soll. Sie muss dann nur in die Programme eingebunden werden.

Beispiel

DSCEXT-Fehlerausgang innerhalb eines COBOL-Programms:

```
DSCEXT.
    ENTRY "DSCEXT".
    DISPLAY "DSCEXT=FEHLERAUSGANG" UPON T.
    STOP RUN.
```

6.7 Umsetztabelle für eine anwendungsspezifische Sortierung

Wenn Sie eine andere, als die in UDS/SQL voreingestellte Sortierung von Character-Feldern (gemäß EBCDI-Code) wünschen, können Sie eine Umsetztabelle erstellen. Die Gültigkeit dieser Tabelle erstreckt sich auf alle Datenbanken der UDS-Session.

Diese Sortierreihenfolge wird nur beim Sortieren der Ergebnismenge einer Suchfrage wirksam, nicht beim Einspeichern von Daten oder bei der Ermittlung von Dateninhalten.

Nationale Felder (Unicode: UTF-16, PICTURE N, USAGE NATIONAL) und numerische Felder sind von einer vorhandenen Umsetztabelle nicht betroffen.

Umsetztabelle erstellen

Sie erstellen ein Assembler-Modul mit dem Namen UDSTRTAB und tragen es in einer beliebigen Bibliothek der Konfigurationskennung ein. Diese Bibliothek muss vor dem Start des independent oder des linked-in DBH zugewiesen werden (siehe hierzu „Umsetztabelle zuweisen“ auf Seite 128).

Um ein eindeutiges Alphabet zu bekommen, müssen Sie darauf achten, dass keine abdruckbaren Zeichen doppelt vorkommen. Dies wird von UDS/SQL nicht überprüft, führt aber zu einer nicht eindeutigen Sortierfolge. Eine nicht eindeutige Sortierfolge kann aber auch beabsichtigt sein, z.B. bei Umlauten, siehe Beispiel.

Es gibt verschiedene Möglichkeiten, eine solche Tabelle zu erstellen:

1. Zeichenumstellung

   Sie gehen von der identischen Umsetztabelle aus (X’000102 ... FF’). Bei nur wenigen Umstellungen ist der Programmieraufwand gering.

   Beispiel

   Die Tabelle kennt nur die 26 Großbuchstaben (X’C1’ bis X’E9’) und Ö auf X’8C’. In der Tabelle wird Ö zwischen O und P eingeordnet.

```
UDSTRTAB START
         TITLE ' UDS-TRANSLATION-TABLE'
TAB      DC    256AL1 (*-TAB)      Identische Umsetztabelle erzeugen
*
         ORG   TAB+X'8C'
         DC    X'D7'               'Ö' -> 'P'
*
         ORG   TAB+X'D7'           'P' -> 'Q'       'Q' -> 'R'
         DC    X'D8D9DA8C'         'R' -> X'DA'   X'DA' -> 'Ö'
         END
```

2. Direkte Angabe des gewünschten Alphabets

Beispiel

Bei der Sortierung wird nicht zwischen Groß- und Kleinbuchstaben unterschieden. Die Umlaute werden hinter den entsprechenden Buchstaben einsortiert, z.B. Ä hinter A, ß hinter s.

```
UDSTRTAB START
         TITLE ' UDS-TRANSLATION-TABLE'
TAB      DC    x'00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F'
               x'10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F'
               x'20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F'
               x'30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F'
               x'40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F'
               x'50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F'
               x'60 61 62 63 64 65 66 E3 68 69 6A 6B 6C 6D 6E 6F'
               x'70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F'
               x'80 C1 C3 C4 C5 C6 C7 C8 C9 CA CA C2 D7 E6 CE CF'
               x'90 D1 D2 D3 D4 D5 D6 D8 D9 DA DA DB DC DD DE DF'
               x'A0 A1 E2 E4 E5 E7 E8 E9 EA EB AA C2 D7 E6 AE AF'
               x'B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF'
               x'C0 C1 C3 C4 C5 C6 C7 C8 C9 CA CA CB CC CD CE CF'
               x'D0 D1 D2 D3 D4 D5 D6 D8 D9 DA DA DB DC DD DE DF'
               x'E0 E1 E2 E4 E5 E7 E8 E9 EA EB EA EB EC ED EE EF'
               x'F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF'
         END
```

Die Leerzeichen sind in der obigen Darstellung nur enthalten, damit das Beispiel leichter lesbar ist, in einer Originalumsetztabelle dürfen sie nicht enthalten sein.

Umsetztabelle zuweisen

Vor dem Aufruf des independent DBH oder des linked-in Anwenderprogramms muss die Bibliothek zugewiesen werden, aus der das selbst erstellte UDSTRTAB-Modul nachgeladen werden soll.

1. Das UDSTRTAB-Modul kann in der Konfigurationskennung in einer PLAM-Bibliothek beliebigen Namens abgespeichert werden. Für die Zuweisung dieser Bibliothek mit dem ADD-FILE-LINK-Kommando steht der Linkname $UDSKONF zur Verfügung.

   ```
   /ADD-FILE-LINK LINK-NAME=$UDSKONF,FILE-NAME=UDSTRTAB-bibliothek
   ```

   Dies ist die von uns empfohlene Standardmethode.

2. Wenn der Linkname $UDSKONF nicht verwendet wird oder das Nachladen nach Methode 1 nicht erfolgreich war, wird aus Kompatibilitätsgründen geprüft, ob eine Bibliothek namens UDS.MODLIB in der Konfigurationskennung existiert. Aus dieser bzw. einer mit SET-TASKLIB zugewiesenen Bibliothek wird dann nachgeladen.

3. Wenn das UDSTRTAB-Modul nach den Methoden 1 und 2 nicht nachgeladen werden konnte, wird das UDSTRAB-Modul aus der im SCI verwalteten SYSLNK-Bibliothek des Produkts UDS/SQL (EBCDIC-Sortierfolge) verwendet.

Wenn das UDSTRTAB-Modul nicht nachgeladen werden kann, wird vom DBH intern die EBCDIC-Sortierfolge genutzt.

7 Nachschlageteil der COBOL-DML

Dieser Abschnitt beschreibt die COBOL-DML-Anweisungen einschließlich ihrer Syntaxregeln.

Die COBOL-DML-Anweisungen geben Sie in COBOL-Programmen an. Der COBOL-Compiler COBOL85 oder COBOL2000 übersetzt sie. Für die COBOL-DML-Anweisungen gelten die gleichen Regeln und die gleichen reservierten Wörter wie für die COBOL-Anweisungen (siehe Handbuch „COBOL-Compiler, Sprachbeschreibung“).

Für die Bearbeitung von Datenbanken mit 4KB- oder 8KB-Seitenlänge benötigen Sie COBOL2000 oder COBOL85 ab der Version 2.2C21, wenn das Anwenderprogramm neu übersetzt werden soll. Dies gilt auch für die Bearbeitung von Datenbanken mit 2048 byte Seitenlänge, wenn Ihr Anwenderprogramm Subschemata verwendet, die **nicht** mit der DDL-Compiler-Option „SUBSCHEMA FORM IS OLD“ übersetzt wurden (siehe Handbuch „Aufbauen und Umstrukturieren“).

7.1 Allgemeine Regeln

Die folgenden Variablen müssen Sie bei der Benutzung eines Formats durch einen aktuellen Wert ersetzen. Dabei sind drei Kategorien von Variablen zu unterscheiden:

variable	**aktueller Wert**
schemaname *subschemaname* *realmname* *setname* *satzname*	Diese Namen müssen aus dem Subschema stammen.
satzelementname *feldname*	Aus dem Zusammenhang in der DML-Anweisung ergibt sich, ob der Name aus der SUB-SCHEMA SECTION stammen muss oder zu einer anderen SECTION gehören kann. Die Namen dürfen nicht indiziert werden.
literal	Das Literal muss in Anführungszeichen (im Fall COBRUN QUOTE1 in Apostrophe) (siehe Handbuch „COBOL-Compiler, Benutzerhandbuch“) eingeschlossen werden. Diese zählen nicht zum Wert des Literals.
ganzzahl	setzt sich aus maximal 15 Ziffern zusammen

Tabelle 18: Metavariablen der COBOL-DML

Sie können das COBOL-Programm mit den COBOL-DML-Anweisungen in beliebigem COBOL-Referenzformat schreiben. Insbesondere entfallen im Free-Form-Referenzformat die bisherigen Spaltenkonventionen.

7.2 Identifikationsteil (ID DIVISION)

Ist das Subschema mit einer PRIVACY LOCK FOR COMPILE-Klausel geschützt, müssen Sie im COBOL-DML-Programm den entsprechenden PRIVACY KEY angeben.

```
[PRIVACY. PRIVACY KEY FOR COMPILE IS literal.]
```

Diese Angabe muss auf die PROGRAM-ID-Angabe folgen.

literal muss eines der Kennwörter sein, die in der IDENTIFICATION DIVISION des Subschemas vergeben wurden.

7.3 Datenteil (DATA DIVISION)

SUB-SCHEMA SECTION

Der Datenteil eines COBOL-Programms, das mit der COBOL-DML arbeitet, muss ein Kapitel SUB-SCHEMA SECTION enthalten.

```
SUB-SCHEMA SECTION.
```

Dieses Kapitel muss das letzte des Datenteils sein. Es enthält die DB-Klausel.

Die DB-Klausel benennt ein Subschema und sorgt dafür, dass alle Bereiche, die für den Informationsaustausch mit dem DBH notwendig sind, in der UWA (User Work Area) reserviert werden.

Die UWA besteht aus folgenden Bereichen:

- SYSTEM-COMMUNICATION-LOCATIONS mit den COBOL-Sonderregistern
- Satzbereich mit
  - je einem eigenen Bereich für jede Satzart des Subschemas
  - den IMPLICITLY-DEFINED-DATA-NAMES (Bereich für Felder, die nicht zu den Satzarten gehören)
- PRIVACY-RECORD mit den BPRIVACY-Schlössern

DB-Eintrag

```
DB subschemaname WITHIN schemaname.
```

subschemaname
: muss der Name eines Subschemas zum angegebenen Schema sein und dem DB-System bekannt sein.

schemaname
: muss der Name eines dem DB-System bekannten Schemas sein.

Die DB-Klausel muss genau einmal vorkommen in jedem Modul, das DML-Anweisungen enthält.

COBOL-DML-Programme, die aus verschiedenen Modulen zusammengebunden sind, können in ihren DB-Klauseln ein Subschema oder auch verschiedene Subschemata zu einem oder verschiedenen Schemata angeben.

Sobald erstmals ein COBOL-Modul mit DB-Klausel durchlaufen wird, wird die Kommunikation mit dem DBH eröffnet, auch wenn keine DML-Anweisung aufgerufen wird.

7.4 Prozedurteil (PROCEDURE DIVISION)

Die Beispiele für die DML-Anweisungen beziehen sich auf eine Beispieldatenbank. Das Schema dieser Datenbank ist im Anhang ab Seite 385 abgebildet.

7.4.1 Übersicht über die COBOL-DML-Anweisungen

Anweisung	Funktion
Format 1: `ACCEPT feldname-1 FROM[ {satzname | setname | realmname} ] CURRENCY`	überträgt den Database-Key-Wert des CRR, CRS, CRA bzw. CRU in das Feld *feldname-1*
Format 2: `ACCEPT feldname-2 FROM[ {satzname | setname | feldname-3} ] REALM-NAME`	überträgt den Namen des Realm, in dem der CRR, CRS, CRU bzw. der zu einem angegebenen Database-Key-Wert gehörige Satz gespeichert ist
`CONNECT[ satzname] TO {setname-1,... | ALL}` `[ RETAINING CURRENCY FOR {setname-2,... | SETS} ]`	hängt den CRU in Set-Occurrences ein
Format1: `DISCONNECT[ satzname] FROM {setname,... | ALL}`	löst den CRU aus Set-Occurrences
Format 2: `DISCONNECT ALL FROM setname,...`	entfernt alle Sätze aus dynamischen Sets
`ERASE satzname[ {PERMANENT | SELECTIVE | ALL} MEMBERS]`	löscht den CRU ggf. mit zugehörigen Membersätzen aus der Datenbank

Tabelle 19: Anweisungen der COBOL-DML (Teil 1 von 6)

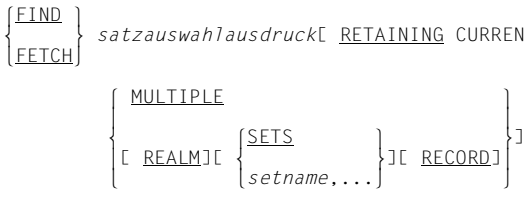
Anweisung	Funktion
	wählt einen oder mehrere Sätze aus der Datenbank aus, abhängig vom Format des Satzauswahlausdrucks und macht den ausgewählten Satz zum CRU und, wenn keine entsprechende RETAINING-Angabe gemacht wird, zum – CRR, – CRS in allen Sets, in denen er Owner oder Member ist, – CRA des Realm, in dem er gespeichert ist. FETCH überträgt den ausgewählten Satz zusätzlich in die UWA des Anwenderprogramms.
Formate des Satzauswahlausdrucks: Format 1:  Format 2:  Format 3: 	Zugriff über den Database Key Zugriff über einen CALC-Key (Hashverfahren) Zugriff auf einen Satz, der in bestimmten Feldinhalten mit dem CRR bzw. CRS übereinstimmt oder Zugriff auf einen Satz, der einem vorhergehend abgearbeiteten Suchausdruck (FIND/FETCH-7) genügt

Tabelle 19: Anweisungen der COBOL-DML (Teil 2 von 6)

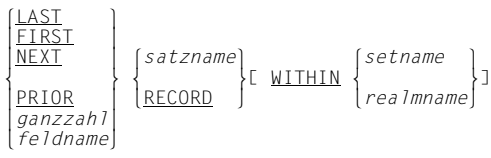
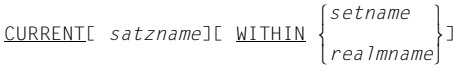
Anweisung	Funktion
Format 4:  	Zugriff auf den letzten oder ersten Satz, auf den Nachfolger oder Vorgänger des CRR, CRS bzw. CRA oder auf einen Satz, dessen Position einem anzugebenden Zahlenwert entspricht innerhalb einer Auswahlmenge. Die Auswahlmenge kann eine Satzart, eine Set-Occurrence, ein Realm oder die Durchschnittsmenge einer Satzart miteinem Realm sein.
Format 5:  	Zugriff auf den CRR, CRS, CRA bzw. CRU
Format 6:  	Zugriff auf den Ownersatz eines CRS

Tabelle 19: Anweisungen der COBOL-DML (Teil 3 von 6)

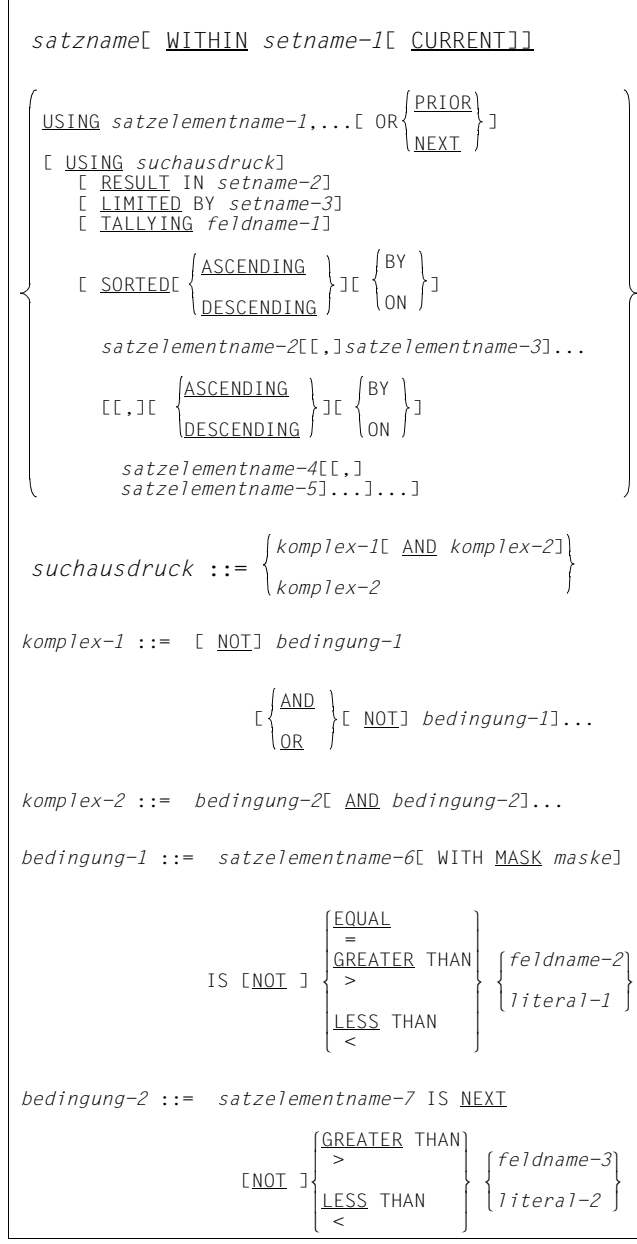
Anweisung	Funktion
Format 7:  	Zugriff auf Sätze über beliebige Felder, ggf. Zählen und Zwischenspeichern der Treffersätze und Suchen mit Maske

Tabelle 19: Anweisungen der COBOL-DML (Teil 4 von 6)

Anweisung	Funktion
FINISH[WITH CANCEL]	beendet eine Transaktion und gibt gesperrte Realms und Seiten frei.
FREE[ALL]	beendet die Wirkung des KEEP-Status.
GET[{ *satzname* / *satzelementname,...* }]	stellt den CRU oder einzelne Felder des CRU im Satzbereich der UWA zur Verfügung.
Format 1: IF[NOT][*setname*] { OWNER / MEMBER / TENANT } { *anweisung-1* / NEXT SENTENCE }[ELSE { *anweisung-2* / NEXT SENTENCE }].	prüft im Programm Set-Mitgliedschaften.
Format 2: IF *setname* IS[NOT] EMPTY { *anweisung-1* / NEXT SENTENCE }[ELSE { *anweisung-2* / NEXT SENTENCE }].	
KEEP	schützt den CRU vor dem Zugriff durch andere Transaktionen bis zu einer FREE-Anweisung oder dem Ende der Transaktion.
MODIFY { *satzname* / *satzelementname,...* } [{ INCLUDING / ONLY } { ALL / *setname-1,...* } MEMBERSHIP] [RETAINING CURRENCY FOR { SETS / *setname-2,...* }]	ändert Feldinhalte des CRU oder hängt ihn innerhalb eines Set in eine andere Set-Occurrence um.
READY[*realmname,...*] [USAGE-MODE IS [{ EXCLUSIVE / PROTECTED }] { RETRIEVAL / UPDATE }]	eröffnet eine Transaktion oder eine Verarbeitungskette.
SET *feldname-1,...*TO *feldname-2*	überträgt den Inhalt eines Database-Key-Feldes in ein oder mehrere Database-Key-Felder.

Tabelle 19: Anweisungen der COBOL-DML (Teil 5 von 6)

Anweisung	Funktion
STORE *satzname*[RETAINING CURRENCY FOR {MULTIPLE [REALM][{SETS *setname*,...}] [RECORD]}]	überträgt einen Satz aus der UWA als neuen Satz in die Datenbank. fügt den neuen Satz in alle Sets ein, für die seine Satzart im Schema als AUTOMATIC Member definiert ist. richtet eine neue Set-Occur rence für jeden Set ein, für den die Satzart im Schema als Ownersatzart definiert ist.
USE FOR DATABASE-EXCEPTION[ON {OTHER *literal*,...}].	definiert Befehlsfolgen, die durchlaufen werden, wenn eine DML-Anweisung mit einem Datenbanksonderzustand endet.

Tabelle 19: Anweisungen der COBOL-DML (Teil 6 von 6)

7.4.2 Anweisungen der COBOL-DML

ACCEPT

Die ACCEPT-Anweisung stellt den Inhalt der angegebenen Currency-Information oder den zu einem Current Record gehörenden Realm-Namen zur Verfügung. Es gibt zwei Formate.

Format 1:

Im Format 1 ermittelt ACCEPT den Database-Key-Wert des CRR, CRS, CRA oder CRU und liefert ihn in einem COBOL-Feld vom Typ USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG ab.

```
ACCEPT feldname-1 FROM[ {satzname } ] CURRENCY
                       {setname   }
                       {realmname }
```

---

feldname-1
: muss mit USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG definiert sein.

  Wenn der CRR, CRS, CRA bzw. CRU einen Database-Key-Wert mit einer REC-REF > 254 und/oder einer RSQ > 2^{24}-1 enthält, muss *feldname-1* mit USAGE IS DATABASE-KEY-LONG definiert sein. Andernfalls überträgt UDS/SQL den Wert 0 in *feldname-1* und gibt den DATABASE-STATUS 15102 aus.

satzname, *setname*, *realmname*
: Geben Sie einen Namen an, so wird der Database-Key-Wert des jeweiligen Current Record in *feldname-1* abgeliefert.

  Geben Sie keinen der Namen an, so wird der Database-Key-Wert des CRU in *feldname-1* abgeliefert. In dieser Variante läuft die Anweisung ohne DBH-Kontakt ab und ist deshalb besonders schnell.

  Ist der angegebene Current Record nicht bekannt, so wird der Wert Null in *feldname-1* abgeliefert.

Beispiel 1

```
       WORKING-STORAGE SECTION.
       01  DB-KEY                          USAGE IS DATABASE-KEY.
       01  DB-KEY-RED          REDEFINES DB-KEY.
           02 REC-REF                      PIC   X.
           02 RSQ                          PIC XXX.
       01  SATZART-NR.
           02 FILLER                       PIC   X.
           02 REC-REF                      PIC   X.
       01  SATZART-NR-RED      REDEFINES SATZART-NR
                                        PIC S9(4) COMP.
       01  SATZFOLGE-NR.
           02 FILLER                       PIC   X.
           02 RSQ                          PIC XXX.
       01  SATZFOLGE-NR-RED    REDEFINES SATZFOLGE-NR
                                        PIC S9(9) COMP.
       01  DB-KEY-AUSGABE.
           02 REC-REF                      PIC 9(3).
           02 RSQ                          PIC 9(8).
      *
           FIND 4 ARTIKEL.
           ACCEPT DB-KEY FROM CURRENCY.
           MOVE REC-REF OF DB-KEY-RED TO REC-REF OF SATZART-NR.
           MOVE RSQ OF DB-KEY-RED TO RSQ OF SATZFOLGE-NR.
           MOVE SATZART-NR-RED TO REC-REF OF DB-KEY-AUSGABE.
           MOVE SATZFOLGE-NR-RED TO RSQ OF DB-KEY-AUSGABE.
           DISPLAY "DB-KEY = " DB-KEY-AUSGABE UPON TERMINAL.
*****************************************************************
(OUT)    DB-KEY = 00900000004
```

Ein DATABASE-KEY-Feld ist ein 4 byte langes Binärfeld zur Speicherung von Database-Key-Werten mit einer REC-REF ≤ 254 und einer RSQ ≤ 2^{24}-1. Bei ACCEPT wird ein DATABASE-KEY-Feld von UDS/SQL wie folgt versorgt: Nach der Ausführung von ACCEPT enthält das erste Byte des Feldes die Satzartnummer (REC-REF) und die übrigen 3 Bytes enthalten die Satzfolgenummer (RSQ). Deshalb ist die Aufteilung des DATABASE-KEY-Feldes in REC-REF und RSQ notwendig. Da COBOL keine 1 byte bzw. 3 byte langen Binärfelder kennt, muss die Satzartnummer zuerst in ein Binärfeld von Halbwortlänge, die Satzfolgenummer in ein Binärfeld von Wortlänge eingespeichert werden. Hierbei werden die beiden Werte in die binäre Darstellungsart konvertiert. Die Satzart ARTIKEL trägt die Satzartnummer 9. Der 4. Satz dieser Satzart trägt die Satzfolgenummer 4.

Beispiel 2

```
       WORKING-STORAGE SECTION.
       01  DB-KEY                          USAGE IS DATABASE-KEY-LONG.
       01  DB-KEY-RED          REDEFINES DB-KEY.
           02 REC-REF                      PIC S9(4) COMP.
           02 FILLER                       PIC S9(4) COMP.
           02 RSQ                          PIC S9(9) COMP.
       01  DB-KEY-AUSGABE.
           02 REC-REF                      PIC 9(5).
           02 RSQ                          PIC 9(10).
      *
           FIND 4 ARTIKEL.
           ACCEPT DB-KEY FROM CURRENCY.
           MOVE REC-REF OF DB-KEY-RED TO REC-REF OF DB-KEY-AUSGABE.
           MOVE RSQ OF DB-KEY-RED TO RSQ OF DB-KEY-AUSGABE.
           DISPLAY "DB-KEY = " DB-KEY-AUSGABE UPON TERMINAL.
*****************************************************************
(OUT)      DB-KEY = 000090000000004
```

Ein DATABASE-KEY-LONG-Feld ist ein 8 byte langes Binärfeld zur Speicherung von Database-Key-Werten.
Bei ACCEPT wird ein DATABASE-KEY-LONG-Feld von UDS/SQL wie folgt mit einem Database-Key-Wert versorgt:

– Byte 1-2: Satzartnummer (REC-REF)

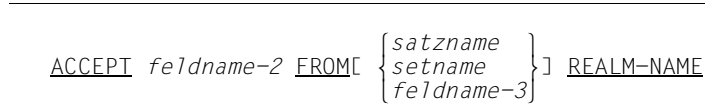
– Byte 3-4: wird nicht verwendet

– Byte 5-8: Satzfolgenummer (RSQ)

Deshalb ist die Aufteilung des DATABASE-KEY-LONG-Feldes in REC-REF und RSQ notwendig.

Format 2:

Im Format 2 ermittelt ACCEPT den Realm, zu dem der in der FROM-Klausel spezifizierte Satz gehört.



feldname-2
muss ein alphanumerisches Feld sein.

feldname-3
muss mit USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG definiert sein.
Wenn Sie für einen Satz, dessen Database-Key-Wert eine REC-REF > 254 und/oder eine RSQ > 2^{24}-1 enthält, den zugehörigen Realm bestimmen wollen, muss *feldname-3* mit USAGE IS DATABASE-KEY-LONG definiert sein.

satzname, *setname*, *feldname-3*
Geben Sie *satzname* an, so wird in *feldname-2* der Name des Realm abgeliefert, zu dem der CRR der Satzart *satzname* gehört.

Geben Sie *setname* an, so wird in *feldname-2* der Name des Realm abgeliefert, zu dem der CRS des Set *setname* gehört.

Geben Sie *feldname-3* an, so wird in *feldname-2* der Name des Realm abgeliefert, zu dem der Satz mit dem in *feldname-3* angegebenen Database-Key-Wert gehört.

Geben Sie keinen der Namen an, so wird in *feldname-2* der Name des Realm abgeliefert, zu dem der CRU gehört.

Existiert der angegebene Satz nicht oder wurde der Satz gelöscht, so werden in *feldname-2* Leerzeichen abgeliefert.

CONNECT

Die CONNECT-Anweisung sorgt dafür, dass der CRU Member in einem oder mehreren Sets wird, in denen seine Satzart wahlfreier Member (d.h. MANDATORY MANUAL, OPTIONAL AUTOMATIC oder OPTIONAL MANUAL Member) ist. CRA und CRR werden nicht geändert.

```
CONNECT[ satzname] TO { setname-1,... }
                      { ALL           }

      [ RETAINING CURRENCY FOR { setname-2,... } ]
                               { SETS          }
```

satzname
: Geben Sie *satzname* an, so muss die Satzart *satzname* zum wahlfreien Member des/der angegebenen Sets erklärt sein. Der CRU muss zu dieser Satzart gehören. Geben Sie *satzname* nicht an, so wird aus der Beschreibung des ersten Set die Membersatzart ermittelt. Für diese Satzart werden dieselben Prüfungen vorgenommen wie für einen angegebenen Satznamen (diese Regel gilt nicht, wenn ALL angegeben ist).

setname-1,...
: Bei dieser Angabe wird der CRU in die entsprechende Set-Occurrence jedes angegebenen Setnamens eingefügt. Die Set-Occurrence wird durch den CRS bestimmt. Handelt es sich um einen dynamischen Set, muss der CRU dieselbe Satzart haben wie die aktuellen Membersätze dieses Set. Er darf keinen Ergebnis-Set einer FIND-7-Anweisung darstellen.

ALL
: Bei dieser Angabe muss die Satzart *satzname* zur Membersatzart mindestens eines Set erklärt sein, der zum Subschema gehört. Der angegebene Satz wird in eine Set-Occurrence aller Sets eingefügt, in denen seine Satzart wahlfreie Membersatzart ist und in denen er zur Zeit nicht Member ist. Dynamische Sets sind bei ALL ausgeschlossen.

RETAINING
: Mit dieser Angabe unterdrücken Sie die Änderung der Currency-Information für *setname-2*,.... Geben Sie SETS an, bleibt die Currency-Information aller betroffenen Sets unverändert.
RETAINING gilt automatisch für alle Sets, die nicht Objekt der CONNECT-Anweisung sind.

READY USAGE-MODE UPDATE ist bei CONNECT-Anweisungen immer erforderlich, außer Sie haben nur dynamische Sets angegeben. In diesem Fall ist auch READY USAGE-MODE RETRIEVAL erlaubt.

DISCONNECT

Die DISCONNECT-Anweisung hängt den CRU aus einer oder mehreren Set-Occurrences aus, vorausgesetzt, dass seine Satzart zum wahlfreien Member der angegebenen Sets erklärt ist (Format 1).

Mit DISCONNECT im Format 2 können Sie alle Sätze aus dynamischen Sets entfernen.

Format 1:

```
DISCONNECT[ satzname] FROM { setname,... }
                           { ALL         }
```

satzname

Geben Sie *satzname* an, so muss die Satzart *satzname* zum wahlfreien Member des/der angegebenen Sets erklärt sein und mit der des CRU übereinstimmen. Geben Sie *satzname* nicht an, so wird aus der Beschreibung des ersten Set die Membersatzart ermittelt. Für diese Satzart werden dieselben Prüfungen vorgenommen wie für einen angegebenen Satznamen (diese Regel gilt nicht, wenn ALL angegeben ist).

setname,...

Bei dieser Angabe wird der CRU aus der entsprechenden Set-Occurrence jedes angegebenen Setnamen entfernt, wenn er OPTIONAL Member ist.
Handelt es sich um einen dynamischen Set, muss der CRU dieselbe Satzart haben wie die aktuellen Membersätze dieses Set. Er darf keine Ergebnismenge einer FIND-7-Anweisung darstellen.

ALL Bei dieser Angabe wird *satzname* aus allen Sets des Subschemas entfernt, in denen er zur Zeit Member ist, wenn er OPTIONAL Membersatz ist.

READY USAGE-MODE UPDATE ist bei diesem Format auf jeden Fall erforderlich.

Der CRS bleibt erhalten und ebenso die Möglichkeit, sequenziell zu suchen (FIND/FETCH-4).

Format 2:

```
DISCONNECT ALL FROM setname,...
```

setname,...
die angegebenen Sets müssen dynamische Sets sein; Sie dürfen Ergebnismengen einer FIND-7-Anweisung angeben.

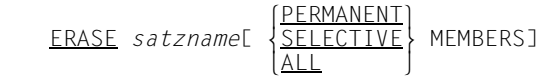
Dieses Format können Sie auch verwenden, wenn das verwendete Subschema mit READY USAGE-MODE RETRIEVAL eröffnet ist.

ERASE

Die ERASE-Anweisung löscht den CRU aus der Datenbank und entfernt ihn aus allen Set-Occurrences, in denen er Member ist. Zusätzlich kann die ERASE-Anweisung

– alle MANDATORY Membersätze des CRU löschen oder

– alle Sätze, die OPTIONAL Member sind in Set-Occurrences, in denen der betreffende Satz der Owner ist, aus diesen Set-Occurrences entfernen und sie wahlweise löschen.



---

satzname

Das Subschema, zu dem dieser Satz gehört, muss enthalten:

– alle Satzarten, von denen als Ergebnis der ERASE-Anweisung Sätze gelöscht oder aus Set-Occurrences entfernt werden,

– alle Sets, in denen irgendein zu löschender Satz der Owner oder ein MANDATORY Member ist.

– alle Sets, aus denen ein Satz entfernt werden muss,

– den Realm, zu dem der Satz, der gelöscht werden soll, gehört,

– alle Realms, die in den WITHIN-Klauseln einer Satzart angegeben sind, die eine Membersatzart des zu löschenden Satzes ist.

Geben Sie nur *satzname* an, wird der CRU nur dann gelöscht, wenn er keine aktuellen Membersätze besitzt.

PERMANENT

Mit dieser Angabe

– löschen Sie den CRU,

– löschen Sie die MANDATORY Membersätze des CRU und

– entfernen Sie die OPTIONAL Membersätze des CRU aus der Set-Occurrence.

Sind MANDATORY Membersätze auch Ownersätze anderer nicht-leerer Set-Occurrences, wirkt die ERASE-Anweisung nach den gleichen Bedingungen auch auf deren Membersätze.

SELECTIVE

Mit dieser Angabe löschen Sie folgende Sätze:

- CRU
- die MANDATORY Membersätze des CRU
- die OPTIONAL Membersätze des CRU, wenn sie nicht in einem weiteren Set einen anderen Owner haben. Sonst werden die OPTIONAL Membersätze nur aus der Set-Occurrence entfernt.

Sind Membersätze auch Ownersätze anderer nicht-leerer Set-Occurrences, wirkt die ERASE-Anweisung nach den gleichen Bedingungen auch auf deren Membersätze.

ALL Mit dieser Angabe löschen Sie den CRU und alle seine Membersätze.

Sind Membersätze auch Ownersätze anderer nicht-leerer Set-Occurrences, wirkt die ERASE-Anweisung nach den gleichen Bedingungen auch auf deren Membersätze.

Wenn Sie PERMANENT, SELECTIVE oder ALL angegeben haben, müssen alle Realms des Subschemas mit READY USAGE-MODE EXCLUSIVE UPDATE ohne Angabe von Realm-Namen eröffnet sein.

Die Einträge in der Currency-Tabelle werden als gelöscht gekennzeichnet. Die Möglichkeit, sequenziell zu suchen (FIND/FETCH-4), bleibt erhalten.

Abhängig von der BMODTT-Einstellung kann der Database-Key-Wert eines gelöschten Satzes nach Ende der löschenden Transaktion grundsätzlich wiederverwendet werden ("REUSE"), oder er bleibt gesperrt ("KEEP").

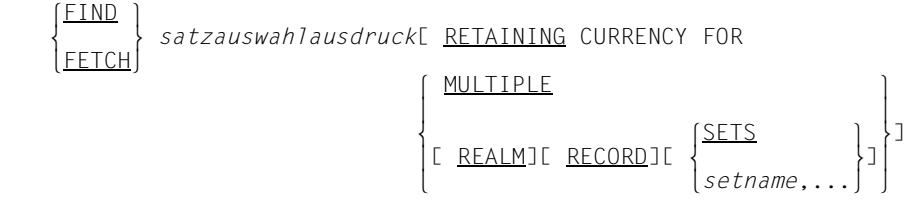
FIND/FETCH

Die FIND-Anweisung wählt einen Satz aus der Datenbank aus und macht ihn zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle, in denen Sie das Aktualisieren nicht unterdrücken. Betroffen sind die Spalten der zugehörigen Satzart, aller Sets, in denen der Satz Owner oder Member ist, und des Realm, in dem der Satz gespeichert ist. Ist der gefundene Satz Member eines dynamischen Set, so wird der betreffende CRS nicht aktualisiert, es sei denn, der dynamische Set ist explizit als WITHIN-Set angegeben.

In einem Fall wählt die FIND-Anweisung auch eine Menge von Sätzen aus und macht den ersten zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle.

Die FETCH-Anweisung führt alle Funktionen der FIND-Anweisung aus. Zusätzlich überträgt sie den neu gewonnenen CRU in die UWA.

Diese zusätzliche Funktion von FETCH wird im Folgenden als bekannt unterstellt und bei der Beschreibung nicht mehr ausdrücklich erwähnt.



---

satzauswahlausdruck
: gibt an, wie Sie auf Sätze zugreifen wollen. Den Satzauswahlausdruck gibt es in 7 Formaten, die auf den folgenden Seiten erklärt sind.

RETAINING
: müssen Sie entweder mit MULTIPLE oder mindestens einer der Angaben REALM, RECORD, SETS oder *setname*,... verwenden.
Sie können damit das Aktualisieren der Currency-Tabelle gezielt dort unterdrücken, wo Sie den Database-Key-Wert des vorangegangenen Satzes bewahren wollen.

  Geben Sie RETAINING... nicht an, so wird der gefundene Satz zum

  – CRA des Realm, in dem er gespeichert ist,
  – CRR seiner Satzart und
  – zum CRS aller Sets, in denen er der Owner oder ein Member ist.

  Currency-Tabellen dynamischer Sets werden nicht aktualisiert, es sei denn, der dynamische Set ist explizit als WITHIN-Set angegeben.

MULTIPLE
unterdrückt bis auf den CRU das Aktualisieren der gesamten Currency-Tabelle.

REALM
unterdrückt das Aktualisieren des CRA.

RECORD
unterdrückt das Aktualisieren des CRR.

SETS unterdrückt das Aktualisieren aller CRS.

setname,…
unterdrückt das Aktualisieren der CRS der angegebenen Sets.

Formate des Satzauswahlausdrucks:

Format 1:

```
[satzname ]DATABASE-KEY IS feldname [OR {PRIOR}]
                                       {NEXT }
```

satzname

Die Anweisung wird nur ausgeführt, wenn der im Feld *feldname* angegebene Database-Key-Wert zur Satzart *satzname* gehört. UDS/SQL wählt dann den zum Database-Key-Wert gehörenden Satz aus der Datenbank aus.

feldname

muss mit USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG definiert sein. In dieses Feld müssen Sie zuvor den Database-Key-Wert des gewünschten Satzes übertragen.
Wenn Sie nach einem Satz suchen wollen, dessen Database-Key-Wert eine REC-REF > 254 und/oder eine RSQ > 2^{24}-1 enthält, muss *feldname* mit USAGE IS DATABASE-KEY-LONG definiert sein.

OR PRIOR

Bei Angabe von OR PRIOR wird der Satz mit dem nächstkleineren Database-Key geliefert, falls es keinen Satz mit dem angegebenen Database-Key gibt.
Der DATABASE-STATUS 04001 zeigt dann an, dass nicht der eigentlich ausgewählte Satz gefunden wurde.
Der DATABASE-STATUS 04024 zeigt in diesem Fall an, dass es weder einen Satz mit dem angegebenen, noch einen Satz mit einem kleineren Database-Key gibt.

OR PRIOR/NEXT kann ab COBOL2000 V1.5 verwendet werden.

OR NEXT

Bei Angabe von OR NEXT wird der Satz mit dem nächstgrößeren Database-Key geliefert, falls es keinen Satz mit dem angegebenen Database-Key gibt.
Der DATABASE-STATUS 04001 zeigt dann an, dass nicht der eigentlich ausgewählte Satz gefunden wurde.
Der DATABASE-STATUS 04024 zeigt in diesem Fall an, dass es weder einen Satz mit dem angegebenen, noch einen Satz mit einem größeren Database-Key gibt.

OR PRIOR/NEXT kann ab COBOL2000 V1.5 verwendet werden.

Beispiel 1

```
       WORKING-STORAGE SECTION.
       01  DB-KEY                          USAGE IS DATABASE-KEY.
       01  DB-KEY-RED          REDEFINES DB-KEY.
           02 REC-REF                      PIC   X.
           02 RSQ                          PIC XXX.
       01  SATZART-NR.
           02 FILLER                       PIC   X.
           02 REC-REF                      PIC   X.
       01  SATZART-NR-RED      REDEFINES SATZART-NR
                                       PIC S9(4) COMP.
       01  SATZFOLGE-NR.
           02 FILLER                       PIC   X.
           02 RSQ                          PIC XXX.
       01  SATZFOLGE-NR-RED    REDEFINES SATZFOLGE-NR
                                       PIC S9(9) COMP.
      *
       FETCH-1.
           MOVE 9 TO SATZART-NR-RED.
           MOVE 15 TO SATZFOLGE-NR-RED.
           MOVE REC-REF OF SATZART-NR TO REC-REF OF DB-KEY-RED.
           MOVE RSQ OF SATZFOLGE-NR TO RSQ OF DB-KEY-RED.
           FETCH DATABASE-KEY IS DB-KEY; DISPLAY "BEZEICHNUNG = "
                 BEZEICHNUNG OF ARTIKEL UPON TERMINAL.
*****************************************************************
(OUT)    BEZEICHNUNG = WEISSBIER
```

Ein DATABASE-KEY-Feld ist ein 4 byte langes Binärfeld zur Speicherung von Database-Key-Werten mit einer REC-REF ≤ 254 und einer RSQ ≤ 2^{24}-1. In ein DATABASE-KEY-Feld muss ein Database-Key-Wert so eingespeichert werden, dass das erste Byte des Feldes die Satzartnummer (REC-REF) enthält und die übrigen drei Bytes die Satzfolgenummer (RSQ) enthalten. Deshalb ist die Aufteilung des DATABASE-KEY-Feldes in REC-REF und RSQ notwendig. Da COBOL keine 1 byte bzw. 3 byte langen Binärfelder kennt, muss die Satzartnummer zuerst in ein Binärfeld von Halbwortlänge, die Satzfolgenummer in ein Binärfeld von Wortlänge eingespeichert werden. Hierbei werden die beiden Werte in die binäre Darstellungsart konvertiert.
Die Ausgabe zeigt den Satz Nr. 15 in der Satzart Nr. 9. Das ist der Satz mit der Bezeichnung WEISSBIER in der Satzart ARTIKEL.

Beispiel 2

```
      WORKING-STORAGE SECTION.
      01  DB-KEY                          USAGE IS DATABASE-KEY-LONG.
      01  DB-KEY-RED          REDEFINES DB-KEY.
          02 REC-REF                      PIC S9(4) COMP.
          02 FILLER                       PIC S9(4) COMP.
          02 RSQ                          PIC S9(9) COMP.
             *
      FETCH-1.
          MOVE 9 TO REC-REF OF DB-KEY-RED.
          MOVE 0 TO FILLER OF DB-KEY-RED.
          MOVE 15 TO RSQ OF DB-KEY-RED.
          FETCH DATABASE-KEY IS DB-KEY; DISPLAY "BEZEICHNUNG = "
                BEZEICHNUNG OF ARTIKEL UPON TERMINAL.
*****************************************************************
(OUT)     BEZEICHNUNG = WEISSBIER
```

Ein DATABASE-KEY-LONG-Feld ist ein 8 byte langes Binärfeld zur Speicherung von Database-Key-Werten.
In ein DATABASE-KEY-LONG-Feld muss ein Database-Key-Wert wie folgt eingespeichert werden:

– Byte 1-2: Satzartnummer (REC-REF)

– Byte 3-4: muss den Wert 0 enthalten

– Byte 5-8: Satzfolgenummer (RSQ)

Deshalb ist die Aufteilung des DATABASE-KEY-LONG-Feldes in REC-REF, Filler und RSQ notwendig.

Format 2:

```
{ANY      }
{         } satzname
{DUPLICATE}
```

ANY wählt einen Satz der Satzart *satzname* zu einem vorgegebenen CALC-Key-Wert aus, indem dieser Wert durch ein Hashverfahren in eine relative Seitennummer umgerechnet wird.
Sie müssen zuvor die Felder, die im Schema als CALC-Key definiert sind (siehe Handbuch „Entwerfen und Definieren“, LOCATION MODE-Klausel) mit dem CALC-Key-Wert des gewünschten Satzes versorgen.

DUPLICATE
ist nur erlaubt, wenn der CALC-Key der Satzart *satzname* im Schema mit DUPLICATES ARE ALLOWED definiert ist.
In dieser Satzart sucht UDS/SQL dann einen vom CRR verschiedenen Satz, der denselben CALC-Key-Wert besitzt wie der CRR und im Realm des CRR liegt. Für die Suche benutzt UDS/SQL das zugehörige Hashverfahren.

satzname
muss eine Satzart bezeichnen, die im Schema mit LOCATION MODE IS CALC definiert ist (siehe Handbuch „Entwerfen und Definieren“, LOCATION MODE-Klausel).

Wenn die Satzart *satzname* auf mehrere Realms verteilt ist, müssen Sie zuvor auch das AREA-ID-Feld (siehe Handbuch „Entwerfen und Definieren“, WITHIN-Klausel) mit dem Namen des Realm versorgen, in dem sich der gewünschte Satz befindet, wenn die Satzart nicht Membersatzart einer verteilbaren Liste ist.

Wenn im Schema DUPLICATES ARE ALLOWED erklärt ist, kann ein CALC-Key-Wert zu mehreren Sätzen der Satzart gehören.
Alle zu einem CALC-Key-Wert gehörenden Sätze der Satzart innerhalb eines Realm erhalten Sie, wenn Sie den ersten Satz mit ANY, jeden weiteren Satz mit DUPLICATE suchen.

Beispiel

```
       FETCH-2.
           FETCH 9 ARTIKELBESCHR; DISPLAY "ART-NR = " ART-NR OF
                 ARTIKELBESCHR UPON TERMINAL.
           MOVE "KLEIDUNG" TO RLMAUSWAHL-3.
           PERFORM DUPLIKATE-LESEN UNTIL DATABASE-STATUS = 04021.
           DISPLAY "KEIN WEITERES DUPLIKAT VORHANDEN" UPON TERMINAL.
           FETCH ANY ARTIKELBESCHR; DISPLAY "ART-NR = " ART-NR OF
                 ARTIKELBESCHR UPON TERMINAL.
           PERFORM DUPLIKATE-LESEN UNTIL DATABASE-STATUS = 04021.
           DISPLAY "KEIN WEITERES DUPLIKAT VORHANDEN" UPON TERMINAL.
           GO TO FETCH-3.
       DUPLIKATE-LESEN.
           FETCH DUPLICATE ARTIKELBESCHR; IF DATABASE-STATUS IS NOT
                 = 04021 DISPLAY "ART-NR = " ART-NR OF ARTIKELBESCHR
                 UPON TERMINAL.
*****************************************************************
(OUT)    ART-NR = 000005
(OUT)    KEIN WEITERES DUPLIKAT VORHANDEN
(OUT)    ART-NR = 000001
(OUT)    ART-NR = 000002
(OUT)    ART-NR = 000003
(OUT)    ART-NR = 000004
(OUT)    ART-NR = 000005
(OUT)    KEIN WEITERES DUPLIKAT VORHANDEN
```

Hier sollen alle Artikelbeschreibungen aus der Datenbank ausgewählt werden, deren CALC-Key-Wert mit dem der 9. Artikelbeschreibung übereinstimmt.
Das Beispiel zeigt, dass die Duplikate erst dann vollständig gefunden werden, wenn die Suche mit FETCH ANY begonnen wird.

Format 3:

```
DUPLICATE WITHIN { satzname } [ USING satzelementname,...]
                 { setname  }
```

satzname

Wenn Sie keine USING-Angabe machen, ist die Auswahlmenge das Ergebnis des zuletzt programmierten Suchausdrucks.
satzname dürfen Sie dann nur angeben, wenn der Suchausdruck auf Satzartebene, also in der Form

```
{FIND  } satzname[ USING suchausdruck]
{FETCH }
```

programmiert wurde (siehe FIND/FETCH-7 auf Seite 164).
satzname muss dann der dort verwendete Satzname sein und UDS/SQL liefert einen Satz aus dem Auswahlergebnis. Der Suchausdruck wird dabei erneut überprüft, sodass auch bei zwischenzeitlichen Änderungen aus parallel laufenden Transaktionen nur die aktuell gültigen Treffersätze ausgegeben werden.
Wenn Sie USING angeben, ist die Auswahlmenge die genannte Satzart. Aus der Satzart wählt UDS/SQL einen Satz aus, der mit dem CRR in den Feldern übereinstimmt, die Sie durch *satzelementname*,... bestimmen.

setname

Wenn Sie keine USING-Angabe machen, ist die Auswahlmenge das Ergebnis des zuletzt programmierten Suchausdrucks.
setname dürfen Sie dann nur angeben, wenn der Suchausdruck auf Setebene, also in der Form

```
{FIND  } WITHIN setname[ USING suchausdruck]
{FETCH }
```

programmiert wurde (siehe FIND/FETCH-7 auf Seite 164).
setname muss dann der dort verwendete Setname sein und UDS/SQL liefert einen Satz aus dem Auswahlergebnis. Der Suchausdruck wird dabei erneut überprüft.
Wenn Sie USING angeben, ist die Auswahlmenge die Set-Occurrence, die den CRS des genannten Set enthält. Aus dieser Set-Occurrence wählt UDS/SQL einen Satz aus, der mit dem CRS in den Feldern übereinstimmt, die Sie durch *satzelementname*,... bestimmen.

satzelementname, ...

müssen Sie genau dann angeben, wenn Sie sich nicht auf eine vorhergehende Anweisung der Form

```
{FIND }
{     } ..... [ USING suchausdruck]
{FETCH}
```

beziehen.
Sie zählen die Satzelemente auf, in denen der gesuchte Satz mit dem CRR der Satzart *satzname* bzw. mit dem CRS des Set *setname* übereinstimmen soll. Der CRS muss dabei ein Membersatz sein.
Die Satzelemente müssen zur Satzart *satzname* bzw. zur Membersatzart des Set *setname* gehören.

Wenn Sie genau ein Satzelement bzw. eine Kombination von Satzelementen angeben, die im Schema als Schlüssel definiert ist, benutzt UDS/SQL die vorhandenen Direktzugriffspfade. Diese Direktzugriffspfade können sein:

– auf Satzartebene:
  Satz-SEARCH-Key-Tabelle oder Hashverfahren für SEARCH-Key
– für den Primärschlüssel auf Setebene:
  Adressliste, Liste oder Sort-Key-Tabelle
– für einen Sekundärschlüssel auf Setebene:
  Set-SEARCH-Key-Tabelle oder Hashverfahren

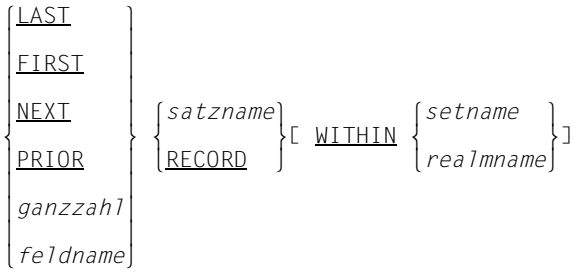
Voraussetzung ist, dass der Schlüssel im Schema mit DUPLICATES ARE ALLOWED definiert ist.
Wenn Sie andere Satzelemente angeben, durchsucht UDS/SQL die Satzart sequenziell.
Wenn Sie genau die Sätze lesen wollen, die in dem Auswahlergebnis eines Suchausdrucks enthalten sind, müssen Sie dies durch eine Folge von FIND-3-Anweisungen tun.

Beispiel

```
       FETCH-3.
           FETCH 2 BESTELLUNG; DISPLAY "BEST-NR = " BEST-NR
                 "     DATUM = " BEST-TAG BEST-MONAT BEST-JAHR UPON
                 TERMINAL.
           FETCH DUPLICATE WITHIN ABGEGEBENE-BEST USING BEST-MONAT,
                 BEST-JAHR; DISPLAY "BEST-NR = " BEST-NR
                 "     DATUM = " BEST-TAG BEST-MONAT BEST-JAHR UPON
                 TERMINAL.
*****************************************************************
(OUT)    BEST-NR = 7995     DATUM = 100182
(OUT)    BEST-NR = 8540     DATUM = 030182
```

Der zweite Satz der Satzart BESTELLUNG trägt das Datum 10.1.82.
Anschließend wird eine Bestellung aus demselben Monat ausgegeben.

Format 4:



LAST liefert den letzten Satz der Auswahlmenge. Die Auswahlmenge bestimmen Sie mit *satzname*, RECORD, *setname* und *realmname*.

FIRST liefert den ersten Satz.

NEXT liefert den Nachfolger

- des CRS bei Angabe von *setname*
- des CRA bei Angabe von *realmname*
- des CRR bei fehlender WITHIN-Angabe.

PRIOR

liefert den Vorgänger

- des CRS bei Angabe von *setname*
- des CRA bei Angabe von *realmname*
- des CRR bei fehlender WITHIN-Angabe.

ganzzahl

muss eine Zahl mit positivem (Standard) oder negativem Vorzeichen sein. 0 ist nicht erlaubt.
UDS/SQL liefert den Satz, dessen Position dem angegebenen Zahlenwert entspricht. UDS/SQL zählt die Position bei

- positiver Ganzzahl vorwärts beim ersten Membersatz beginnend,
- negativer Ganzzahl rückwärts beim letzten Membersatz beginnend.

Die Angabe 1 entspricht daher FIRST, -1 entspricht LAST.
Negative Werte sind nicht erlaubt, wenn der Set mit der SSL als MODE IS CHAIN ohne LINKED TO PRIOR definiert ist.

feldname

muss ein numerisches Feld sein, das eine ganze Dezimalzahl mit Vorzeichen enthält. Für die Zahl gelten die vorstehenden Regeln für *ganzzahl*.

satzname

legt die genannte Satzart als Auswahlmenge fest, wenn Sie keine WITHIN-Angabe anschließen. Die Reihenfolge ist durch aufsteigende Database-Key-Werte bestimmt.

RECORD

dürfen Sie nicht ohne WITHIN verwenden.

setname

legt die Set-Occurrence als Auswahlmenge fest, die den CRS des genannten Set enthält. Die Reihenfolge ist durch die ORDER-Klausel für diesen Set im Schema bestimmt. Der Ownersatz ist in dieser Reihenfolge Vorgänger des ersten Membersatzes.

Wenn *satzname* angegeben ist, muss er die Membersatzart dieses Set bezeichnen.

Da FIND7 mit RESULT IN und SORTED nicht erlaubt ist, kann der Set kein sortierter Ergebnisset sein.

realmname

Wenn *satzname* angegeben ist, muss der Realm-Name aus der DDL-WITHIN-Klausel dieser Satzart stammen. Die Auswahlmenge besteht dann aus den Sätzen, die zur Satzart *satzname* gehören und im Realm *realmname* liegen.
Sonst muss der Realm-Name aus einer DDL-WITHIN-Klausel irgendeiner zum Subschema gehörenden Satzart stammen und die Auswahlmenge besteht dann aus den Sätzen des Realm.

In beiden Fällen ist die Reihenfolge der Sätze durch aufsteigende Database-Key-Werte bestimmt.

Beispiel

```
       FETCH-4.
            FETCH FIRST LIEFERANT; DISPLAY LIEFER-NAME UPON TERMINAL.
            FETCH 2 BESTELLUNG WITHIN ABGEGEBENE-BEST; DISPLAY
                  "BEST-NR = " BEST-NR "     DATUM = " BEST-TAG
                  BEST-MONAT BEST-JAHR UPON TERMINAL.
*****************************************************************
(OUT)    MONA MODEHAUS
(OUT)    BEST-NR = 7995     DATUM = 100182
```

Zunächst wird der erste Satz der Satzart LIEFERANT ausgegeben. Anschließend wird der zweite Satz aus der Set-Occurrence dieses Lieferanten im Set ABGEGEBEN-BEST ausgewählt.

Format 5:

```
CURRENT[ satzname][ WITHIN { setname   }]
                           { realmname }
```

satzname
: bewirkt den Zugriff auf den CRR dieser Satzart und setzt somit die Currency-Information auf den Stand des CRR zurück, wenn Sie keine WITHIN-Angabe anschließen: Der CRR wird wieder zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle bzw. dort, wo Sie das Aktualisieren nicht mit RETAINING unterdrücken.

setname
: bewirkt den Zugriff auf den CRS dieses Set und setzt somit die Currency-Information auf den Stand des CRS zurück: Der CRS wird wieder zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle bzw. dort, wo Sie das Aktualisieren nicht mit RETAINING unterdrücken.

  Wenn *satzname* angegeben ist, muss er die Membersatzart des Set bezeichnen. UDS/SQL führt die Anweisung dann nur aus, wenn der CRS zu dieser Satzart gehört, d.h. wenn der CRS nicht der Ownersatz ist.

realmname
: bewirkt den Zugriff auf den CRA dieses Realm und setzt somit die Currency-Information auf den Stand des CRA zurück: Der CRA wird wieder zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle bzw. dort, wo Sie das Aktualisieren nicht mit RETAINING unterdrücken.

  Wenn *satzname* angegeben ist, muss der Realm-Name aus der DDL-WITHIN-Klausel dieser Satzart stammen. UDS/SQL führt die Anweisung dann nur aus, wenn der CRA zu dieser Satzart gehört.
  Sonst muss der Realm-Name aus der DDL-WITHIN-Klausel irgendeiner zum Subschema gehörenden Satzart stammen.

Wenn Sie FIND CURRENT ohne weitere Angaben programmieren, wird die Currency-Information auf den Stand des CRU gebracht. Diese Anweisung ist nur sinnvoll, wenn Sie zuvor RETAINING benutzt haben.

Beispiel

```
       FETCH-5.
           FIND FIRST LIEFERANT.
           FIND NEXT LIEFERANT RETAINING CURRENCY FOR LIEFERANTEN.
           FETCH CURRENT LIEFERANT RETAINING CURRENCY FOR LIEFERANTEN;
                 DISPLAY LIEFER-NAME UPON TERMINAL.
           FETCH CURRENT LIEFERANT WITHIN LIEFERANTEN; DISPLAY
                 LIEFER-NAME UPON TERMINAL.
*****************************************************************
(OUT)    AUGUSTINERBRAEU
(OUT)    MONA MODEHAUS
```

Das Beispiel zeigt die Wirkung der Currency-Information und der RETAINING-Angabe. Der zuerst ausgegebene Satz ist der CRR der Satzart LIEFERANT. Der zweite ausgegebene Satz stammt zwar aus derselben Satzart, ist jedoch der CRS im Set LIEFERANTEN, in dem LIEFERANT die Membersatzart ist.

Format 6:

```
OWNER WITHIN setname
```

setname

darf keinen SYSTEM-Set bezeichnen, da diese keinen Ownersatz haben. UDS/SQL greift auf den Owner des CRS zu. Wenn der CRS selbst der Ownersatz ist, bedeutet das einen Zugriff auf den CRS.

Beispiel

```
       FETCH-6.
           FETCH CURRENT BESTELLUNG WITHIN ABGEGEBENE-BEST; DISPLAY
                 "BEST-NR = " BEST-NR "     DATUM = " BEST-TAG
                 BEST-MONAT BEST-JAHR UPON TERMINAL.
           FETCH OWNER WITHIN ABGEGEBENE-BEST; DISPLAY "LIEFER-NAME = "
                 LIEFER-NAME UPON TERMINAL.
*****************************************************************
(OUT)    BEST-NR = 7995     DATUM = 100182
(OUT)    LIEFER-NAME = MONA MODEHAUS
```

Hier wird der Owner zur Bestellung 7995 im Set ABGEGEBENE-BEST ausgegeben.

Format 7:

```
satzname[ WITHIN setname-1[ CURRENT]]

        ┌                                                                    ┐
        │ USING satzelementname-1,...[OR {PRIOR}]                            │
        │                                {NEXT }                             │
        │                                                                    │
        │[ USING suchausdruck][ RESULT IN setname-2]                         │
        │                                                                    │
        │      [ LIMITED BY setname-3][ TALLYING feldname-1]                 │
        │                                                                    │
        │      [ SORTED[ {ASCENDING } ][ {BY} ]                              │
        │                {DESCENDING}    {ON}                                │
        │                                                                    │
        │       satzelementname-2[[,]satzelementname-3]...                  │
        │                                                                    │
        │       [[,][ {ASCENDING } ][ {BY} ]                                │
        │             {DESCENDING}    {ON}                                   │
        │                                                                    │
        │       satzelementname-4[[,]satzelementname-5]...]...]              │
        └                                                                    ┘

suchausdruck    ::= {komplex-1[ AND komplex-2]}
                    {komplex-2                }

komplex-1       ::= [NOT ]bedingung-1[ {AND}[ NOT] bedingung-1]...
                                      {OR }

bedingung-1     ::= satzelementname-6[ WITH MASK maske] IS

                                  {EQUAL           }
                                  {  =             }   {feldname-2}
                          [NOT ]  {GREATER THAN    }   {literal-1 }
                                  {  >             }
                                  {LESS THAN       }
                                  {  <             }

komplex-2       ::= bedingung-2[ AND bedingung-2]...

bedingung-2     ::= satzelementname-7 IS NEXT

                                  {GREATER THAN    }   {feldname-3}
                          [NOT ]  {  >             }   {literal-2 }
                                  {LESS THAN       }
                                  {  <             }
```

satzname

legt die genannte Satzart als Auswahlmenge fest, wenn Sie keine WITHIN-Angabe anschließen.

setname-1

legt eine Set-Occurrence des genannten Set als Auswahlmenge fest. Die Auswahl der gewünschten Set-Occurrence bestimmen Sie mit CURRENT.
satzname muss die Membersatzart dieses Set bezeichnen. Im dynamischen Set ist das die Satzart, deren Sätze sich in dem dynamischen Set befinden.

CURRENT

siehe Abschnitt „Set-Occurrence auswählen“ auf Seite 170.

satzelementname-1

müssen Satzelemente der Satzart *satzname* bezeichnen. Sie zählen damit die Satzelemente auf, in denen der gesuchte Satz mit den in der UWA vorgegebenen Werten übereinstimmen soll. Die Satzelemente müssen Sie deshalb zuvor in der UWA mit den gewünschten Werten versorgen. Sie dürfen auch Datengruppennamen verwenden.

OR PRIOR

liefert den Satz mit dem in der UWA vorgegebenen Schlüsselwert.
Wenn es keinen Satz mit dem in der UWA vorgegebenen Schlüsselwert gibt, liefert die Klausel OR PRIOR den gemäß Sortierreihenfolge unmittelbar vorhergehenden Satz der Set-Occurrence. Die Sortierreihenfolge ist festgelegt durch *satzelementname-1,...*, wobei *satzelementname-1,...* der ASCENDING KEY bzw. DESCENDING Key des Set *setname-1* sein muss. Eine Verletzung dieser Bedingungen quittiert UDS/SQL zur Laufzeit mit dem DATABASE-STATUS 04101.

Existieren zu einem Schlüsselwert Duplikatsätze, so gibt UDS/SQL von diesen Sätzen den Satz mit der größten Satzfolgenummer (RSQ) aus. Somit können Sie mit einer Anweisung FIND-7 OR PRIOR und einer Folge von Anweisungen FIND-4 PRIOR sämtliche Duplikate ermitteln.

Wenn der in der UWA vorgegebene Schlüsselwert kleiner ist als der kleinste in der Set-Occurrence vorhandene Schlüsselwert, meldet UDS/SQL den DATABASE-STATUS 04024.

OR NEXT

liefert den Satz mit dem in der UWA vorgegebenen Schlüsselwert.
Wenn es keinen Satz mit dem in der UWA vorgegebenen Schlüsselwert gibt, liefert die Klausel OR NEXT den gemäß Sortierreihenfolge unmittelbar nachfolgenden Satz der Set-Occurrence. Die Sortierreihenfolge ist festgelegt durch *satzelementname-1,...,* wobei eine der beiden folgenden Bedingungen erfüllt sein muss:

– *satzelementname-1,...* ist der ASCENDING KEY, DESCENDING KEY oder ein SEARCH KEY USING INDEX des Set *setname-1*.

– *satzelementname-1,...* ist ein SEARCH KEY USING INDEX der Satzart *satzname*.

Andernfalls meldet UDS/SQL zur Laufzeit den DATABASE-STATUS 04101.

Existieren zu einem Schlüsselwert Duplikatsätze, so gibt UDS/SQL von diesen Sätzen den Satz mit der kleinsten Satzfolgenummer (RSQ) aus. Somit können Sie sämtliche Duplikate ermitteln

– mit einer Anweisung FIND-7 OR NEXT und einer Folge von Anweisungen FIND-4 NEXT oder

– mit einer Anweisung FIND-7 OR NEXT und einer Folge von Anweisungen FIND-3.

Wenn der in der UWA vorgegebene Schlüsselwert größer ist als der größte in der Set-Occurrence vorhandene Schlüsselwert, meldet UDS/SQL den DATABASE-STATUS 04024.

suchausdruck

gibt die Feldinhalte, die die zu suchenden Sätze aufweisen müssen, durch Vergleichsbedingungen vor.
Folgende Besonderheiten können Sie dabei ausnutzen:

– Speichern der gesamten Treffermenge:
UDS/SQL sucht nicht nur einen Satz, sondern es nimmt alle Sätze, die dem Suchausdruck genügen, in einen dynamischen Set auf. Der erste gefundene Satz wird aktueller Satz in der Currency-Tabelle.
– Ungleichungen als Vergleichsbedingung:
UDS/SQL kann alle Sätze auswählen, die einer Ungleichung genügen (*bedingung-1*) oder nur den Satz, der die Ungleichung am genauesten erfüllt (*bedingung-2*).
– logische Operatoren UND, ODER, NICHT als Vergleichsbedingung
– Masken
– Wertvorgabe wahlweise in der UWA (*feldname-2/-3*) oder in der Anweisung selbst (*literal-1/-2*)
– Treffersätze in einem Set ablegen (RESULT)
– nur die Durchschnittsmenge zweier Auswahlmengen durchsuchen (LIMITED)

– Treffersätze zählen (TALLYING)
– Treffersätze sortieren (SORTED BY).

Das verwendete Subschema muss einen Temporären Realm enthalten.

USING

Wenn Sie keine USING-Angabe machen, fasst UDS/SQL dies als Suchausdruck auf, dem alle Sätze der Auswahlmenge genügen.

bedingung-1

liefert alle Sätze, die der Vergleichsbedingung genügen. Negative Feldinhalte und negative Vergleichsinhalte sind nur erlaubt, wenn der Vergleichsoperator "EQUAL" ist..
Eine Aneinanderreihung mehrerer *bedingung-1* wertet UDS/SQL nach folgenden Regeln aus: NOT bindet stärker als AND und AND bindet stärker als OR.
Diese Prioritätenfolge können Sie ändern, indem Sie Verknüpfungen, die UDS/SQL als Einheit auswerten soll, einklammern.

satzelementname-6

bezeichnet ein Satzelement der Satzart *satzname*, das einen vorgegebenen Wert enthalten soll. Es darf kein Feld variabler Länge enthalten.
Sie dürfen auch Datengruppennamen verwenden.
Wenn UDS/SQL nicht den gesamten Inhalt des Satzelementes prüfen soll, können Sie mit einer Maske Teile des Satzelementes ausblenden.
Die Wertvorgabe geschieht durch *feldname-2* oder *literal-1*.

maske blendet Teile von *satzelementname-6* aus. *maske* muss genauso definiert sein wie *satzelementname-6*, andernfalls wird die Maske automatisch in die Form von *satzelementname-2* konvertiert.
Jedes Byte der Maske muss den Wert LOW VALUE oder HIGH VALUE besitzen. Die Positionen in *satzelementname-6*, die in der Maske mit LOW VALUE besetzt sind, wertet UDS/SQL beim Vergleich nicht aus. Diese Positionen müssen auch in *feldname-2* mit LOW VALUE besetzt sein.

feldname-2

kann ein Satzelement, aber auch ein Feld in der WORKING-STORAGE SECTION sein, das die Wertvorgabe für *satzelementname-6* enthält. Es muss genauso definiert sein wie *satzelementname-6*, wenn Sie nicht WITH MASK verwenden. Sonst muss es nur in der Länge mit *satzelementname-6* übereinstimmen.
feldname-2 müssen Sie zuvor mit dem gewünschten Wert versorgen.

literal-1

ist der Wert, den Sie für *satzelementname-6* in der Anweisung vorgeben. *literal-1* muss genauso lang sein wie *satzelementname-6*.

bedingung-2

liefert alle Sätze, die der Vergleichsbedingung genügen. Ungleichungen, negative Feldinhalte und negative Vergleichsinhalte sind nicht erlaubt.

Nachdem zunächst *bedingung-1* in *komplex-1* ausgewertet wurde, werden die Bedingungen *bedingung-2* in *komplex-2* der Reihe nach ausgewertet, beginnend mit der am weitesten links stehenden Bedingung.

Durch den Zusatz NEXT wird die Treffermenge einer *bedingung-2* in *komplex-2* auf diejenigen Sätze der Satzart *satzname* eingeschränkt, die als *satzelement-6* den Wert W mit den folgenden beiden Eigenschaften enthalten:

– W genügt dem Vergleich in *bedingung-2*.

– Von allen in der Datenbank vorkommenden Werten für *satzelement-6*, die dem Vergleich in *bedingung-2* genügen, liegt W am nächsten bei dem durch *literal-2* bzw. *feldname-3* spezifizierten Vergleichswert.

satzelementname-7

bezeichnet ein Satzelement der Satzart *satzname*, das einen vorgegebenen nicht negativen Wert enthalten soll. Es darf kein Feld variabler Länge enthalten.
Sie dürfen auch Datengruppennamen verwenden.
Die Wertvorgabe geschieht durch *feldname-3* oder *literal-2*.

feldname-3

kann ein Satzelement, aber auch ein Feld in der WORKING-STORAGE SECTION sein, das die Wertvorgabe für *satzelementname-7* enthält. Es muss genauso definiert sein wie *satzelementname-7*.
feldname-3 müssen Sie zuvor in der UWA mit dem gewünschten nicht negativen Wert versorgen.

literal-2

ist der Wert, den Sie für *satzelementname-7* in der Anweisung vorgeben. *literal-2* muss genauso lang sein wie *satzelementname-6* und darf nicht negativ sein.

RESULT

brauchen Sie, wenn UDS/SQL die Treffersätze in einen expliziten dynamischen Set speichern soll, den Sie mit weiteren FIND/FETCH-4-Anweisungen ansprechen können. Auf diese Weise können Sie die Suche nach Sätzen so programmieren, dass insgesamt mehrere Hierarchiestufen der Datenstruktur durchlaufen werden, d.h. für die Suche können Sie über Set-Beziehungen insgesamt Felder verschiedener Satzarten benutzen (komplexe Suchfrage).

setname-2

muss einen dynamischen Set bezeichnen.

LIMITED

dürfen Sie nur angeben, wenn *setname-1* keinen dynamischen Set bezeichnet. LIMITED begrenzt die Auswahlmenge auf die Sätze, die

– zur Satzart *satzname* bzw. zur ausgewählten Set-Occurrence des Set *setname-1* gehören und zusätzlich

– im Set *setname-3* enthalten sind.

setname-3 darf kein sortierter dynamischer Set sein, da die Durchschnittbildung einer Treffermenge mit einem sortierten dynamischen Set nicht möglich ist.
Wenn *setname-3* ein unsortierter dynamischer Set ist, dürfen Sie die SORTED-Klausel (s.u.) nicht angeben, da der Durchschnitt einer Treffermenge und eines dynamischen Set nicht sortiert werden kann.

setname-3

muss einen dynamischen Set bezeichnen.

TALLYING

zählt die Sätze, die dem Suchausdruck genügen und speichert die Anzahl ins Feld *feldname-1*.

feldname-1

muss ein numerisches Feld bezeichnen.

SORTED ...

dürfen Sie nicht angeben, wenn *setname-3* (siehe LIMITED-Klausel) ein dynamischer Set ist.
Die SORTED-Klausel sortiert die Sätze, die dem Suchausdruck genügen. Wenn Sie eine eigene Umsetztabelle in die UDS.MODLIB eingefügt haben, wird nach Ihrer eigenen Sortierfolge sortiert.

Bei ASCENDING wird aufsteigend sortiert.

Bei DESCENDING wird absteigend sortiert.

satzelementname-2 bis *satzelemntname-5*

müssen Satzelemente der satzart *satzname* bezeichnen.
Sie zählen damit die Satzelemente auf, nach denen die Sätze sortiert werden, und bestimmen durch die Reihenfolge der Satzelemente die Priorität der Sortierfelder. *satzelementname-2,..5* darf auch ein Datengruppenname sein. Die Sätze werden typgerecht sortiert, sofern ein Subschema der UDS/SQL V2.0 oder jünger vorliegt, das nicht mit „SUBSCHEMA FORM IS OLD“ übersetzt wurde. Datengruppen werden dabei wie ein Character-Feld behandelt. Wenn Sie nichts angeben, wird aufsteigend sortiert.
Standardwert der ersten Angabe ist ASCENDING, Standardwert der in der Wiederholung stehenden Angabe ist die gerade aktuelle Sortierrichtung. Es gilt die Einschränkung, dass alle Sortierfelder die gleiche Sortierrichtung aufweisen müssen.

Set-Occurrence auswählen

Ein Set, der nicht SYSTEM-Set ist, besitzt im Allgemeinen mehrere Set-Occurrences, sodass zunächst die Set-Occurrence bestimmt sein muss, die zu durchsuchen ist.

Wenn Sie CURRENT angeben, ist die Set-Occurrence durch den CRS bestimmt. Dasselbe gilt, wenn im Schema die Auswahlmethode für die Set-Occurrences dieses Set mit SELECTION THRU CURRENT OF SET angegeben ist.

Wenn Sie nicht CURRENT angeben und im Schema ist die Auswahlmethode mit SELECTION THRU LOCATION MODE OF OWNER festgelegt, so müssen Sie nur den Schlüsselwert, der zum Ownersatz der gewünschten Set-Occurrence gehört, in die UWA übertragen.

Der Schlüssel ist

– ein Database Key, wenn die Ownersatzart im Schema mit LOCATION MODE IS DIRECT oder LOCATION MODE IS DIRECT-LONG definiert ist.

– ein CALC-Key, wenn die Ownersatzart im Schema mit LOCATION MODE IS CALC definiert ist.

UDS/SQL wählt die Set-Occurrence aus, indem es über den Database Key bzw. über das Hashverfahren auf den zugehörigen Ownersatz zugreift und ihn automatisch zum CRS macht.

Die Tabelle fasst noch einmal zusammen, wie Sie die Set-Occurrence auswählen müssen:

Schema-Definition / **FIND-Anweisung**	SET OCCURRENCE SELECTION IS THRU		
	CURRENT OF SET	LOCATION MODE OF OWNER	
		LOCATION MODE IS { DIRECT | DIRECT-LONG } { *feldname* | *bezeichner* }	LOCATION MODE IS CALC USING *feldname* ...
ohne CURRENT	CRS	Database-Key-Wert im Feld *feldname* bzw. *bezeichner* in UWA bzw. in den in der ALIAS-Klausel genannten Feldern	CALC-Key-Wert in den Feldern *feldname*,... in der UWA bzw. in den in der ALIAS-Klausel genannten Feldern
mit CURRENT	CRS	CRS	CRS

Tabelle 20: Auswahl der Set-Occurrences

Direktzugriffspfade ausnutzen

Durch Ausnutzen von Direktzugriffspfaden können Sie den Ablauf einer Suchfrage wesentlich beschleunigen. Solche Direktzugriffspfade sind

– Primärschlüssel auf Satzartebene (LOCATION MODE IS CALC)

– Primärschlüssel auf Setebene (ASCENDING/DESCENDING KEY); ausgeklammert ist der Fall ORDER IS SORTED ohne den Zusatz INDEXED (nur bei MODE IS CHAIN möglich)

– Sekundärschlüssel auf Satzart- und Setebene (SEARCH KEY USING INDEX/CALC).

Damit ein Schlüssel in *satzelementname-1*,... ausgewertet wird, muss beim FIND/FETCH-7 Folgendes beachtet werden:

– wenn WITHIN *setname-1* nicht angegeben ist:
Der Schlüssel ist auf Satzartebene definiert

– wenn WITHIN *setname-1* angegeben ist:
Der Schlüssel ist in dem Set *setname-1* definiert.

Damit ein Schlüssel in *suchausdruck* ausgewertet wird, muss beim FIND/FETCH-7 beachtet werden,

– wenn kein WITHIN *setname-1* oder wenn WITHIN *setname-1*, wobei *setname -1* ein SYSTEM-Set der Mitgliedsart MANDATORY AUTOMATIC ist:
Der Schlüssel ist auf Satzartebene definiert oder in einem beliebigen SYSTEM-Set der Mitgliedsart MANDATORY AUTOMATIC, dessen Membersatzart die zu Grunde liegende Satzart ist

– wenn WITHIN *setname-1*, wobei *setname-1* kein SYSTEM-Set der Mitgliedsart MANDATORY AUTOMATIC ist:
Der Schlüssel ist in dem Set *setname-1* definiert.

- Direktzugriff beim FIND-7 USING *satzelementname-1*,...

  Handelt es sich um einen aus mehreren Feldern bestehenden Schlüssel, so müssen *satzelementname-1*,... genau mit den Feldern des zusammengesetzten Schlüssels übereinstimmen, auch in der Reihenfolge.
  Entspricht einem zusammengesetzten Schlüssel genau eine Datengruppe im Subschema, so kann statt *satzelementname -1*,... auch der Datengruppenname angegeben werden. Eine Datengruppe wird dabei wie ein Feld mit TYPE=CHARACTER behandelt.
  Ist *satzelementname-1*,... der Primärschlüssel auf Satzartebene, wird die Tabellensuche nicht unterstützt. In diesem Fall ist ein FIND-2 besser geeignet.

- Direktzugriff beim FIND-7 USING *suchausdruck*

  Bis einschließlich UDS/SQL V2.0 konnte eine Bedingung innerhalb eines Suchausdrucks nur dann mittels eines vorhandenen Direktzugriffspfades („Index“) ausgewertet werden, wenn der betreffende Satzelementname Elementarfeld eines Single Key ist, oder wenn er eine Datengruppe ist, deren Elementarfelder genau die Felder eines Compound Key umfassen.
  Im Fall der Datengruppe bestand zwar die Möglichkeit, Elementarfelder der Datengruppe bezüglich der Bedingung für irrelevant zu erklären, indem die Felder mit Maskenzeichen versehen werden, es blieb aber das Problem der Datenreplikation und Datenunabhängigkeit (Felder müssen zum Beispiel mehrfach im Satz erscheinen, wenn sie in mehreren Datengruppen-Keys vorkommen).

  Ab UDS/SQL V2.2 werden nur die relevanten Felder im Suchausdruck benannt. Unter Beibehaltung des Direktzugriffs wird eine Bedingung ausgewertet, auch wenn sie als Datengruppe oder als Elementarfeld nur einen Teil der Felder des zu Grunde liegenden Compound Keys umfasst.

  Mit „AND“ und „OR“ verknüpfte Maskenbedingungen, die sich auf ein und dieselbe Indextabelle beziehen, werden ab UDS/SQL V2.2 vollständig über Direkzugriffspfade ausgewertet.
  Bis einschließlich UDS/SQL V2.0 wurde bei "AND" nur eine der Bedingungen via Indextabelle ausgewertet und die entstandene Obermenge sequenziell weiterverarbeitet, bei "OR" erfolgte die Tabellensuche ohne Eingrenzung des Suchbereichs.

  Anwendungsszenarien zur Nutzung von Compound Keys

  Die Formulierung von Suchfragen mittels Compound Keys bietet folgende Vorteile:

  – Datenreplikation wird überflüssig, weil Felder nur einmal im Satz erscheinen müssen

  – zusätzliche Search Keys können durch geschickte Auswahl der Felder in Compound Keys vermieden werden

  – Compound Keys gestatten unterschiedliche Feldtypen, während Datengruppen bestimmte Feldtypen verbieten (Datengruppen werden wie Felder vom Typ PIC(X) behandelt und dürfen somit keine nichtabdruckbaren numerischen Datentypen enthalten)

  – Masken werden nicht benötigt, weil irrelevante Daten in der Suchbedingung weggelassen werden können

  Die Auswertung mittels Compound Keys führt also insgesamt zu kürzeren Sätzen und sorgt für Datenunabhängigkeit der Anwendungsprogramme, was einen geringeren Pflege- und Änderungsaufwand bedeutet, zumal Datengruppen bei Änderungen in der Datenstruktur auch ohne Replikatfelder einen höheren Pflegeaufwand erfordern.

Für Anwendungen im SQL-Umfeld, die mit einem ODBC-Server arbeiten, bieten Compound Keys eine ideale Unterstützung, da die ODBC-Schnittstelle keine Datengruppen unterstützt und die Nutzung von Compound Keys hier eine erhebliche Verbesserung darstellt.

Die folgenden Beispiele verdeutlichen die Nutzung des Compound Key ab UDS/SQL V2.2. Allen Beispielen liegt die Satzart KUNDE zu Grunde. K-BRANCHE bezeichnet die Branche, K-PLZ die Postleitzahl und K-VOLUMEN das Gesamtbestellvolumen des Kunden:

```
RECORD NAME IS KUNDE

UDS/SQL V2.0          02       K-BRANCHE-PLZ
                        03     K-BRANCHE                  PIC X(20)
                        03     K-PLZ
                          04   K-REGION                   PIC 9
                          04   K-PLZREST                  PIC 9999
                      02       K-REGION-BRANCHE
                        03     K-REGION-1                 PIC 9
                        03     K-BRANCHE-1                PIC X(20)
                      02       K-VOLUMEN-REGION
                        03     K-VOLUMEN                  PIC 9(9)
                        03     K-REGION-2                 PIC 9


ab UDS/SQL V2.2       02       K-BRANCHE                  PIC X(20)
                      02       K-REGION                   PIC 9
                      02       K-PLZREST                  PIC 9999
                      02       K-VOLUMEN                  TYPE DECIMAL 9
```

K-REGION-1und K-REGION-2 sind Replikate von K-REGION. K-BRANCHE-1 ist ein Replikat von K-BRANCHE.

K-VOLUMEN waren in UDS/SQL V2.0 als abdruckbare numerische Zahl deklariert, weil K-VOLUMEN als Teil einer Datengruppe mit Maskenzeichen bearbeitet werden soll. Ab UDS/SQL V2.2 können Datengruppen und Maskenzeichen bei Nutzung von Compound Keys entfallen und damit kann der Feldtyp frei gewählt werden.

Beispiel 1

Aufgabe sei die Auswertung der Kunden zum einen nach Branchen, zum anderen nach Region. Damit die Auswertung performant abläuft, werden jeweils zwei Search Keys definiert. Dabei kann ab UDS/SQL V2.2 auf Datenreplikation verzichtet werden.

UDS/SQL V2.0:

```
SEARCH KEY IS K-BRANCHE, K-REGION, K-PLZREST USING INDEX
SEARCH KEY IS K-REGION-1, K-BRANCHE-1        USING INDEX

FETCH KUNDE USING K-BRANCHE-PLZ ='ELEKTRO*'     *suche Kunden einer Branche*
FETCH  KUNDE USING K-REGION-BRANCHE ='7*'
```

Das Zeichen '*' steht für Maskenzeichen am Ende eines Bezeichners. Bezüglich COBOL-DML ist die Anweisung syntaktisch inkorrekt.

ab UDS/SQL V2.2:

```
SEARCH KEY IS K-BRANCHE, K-REGION, K-PLZREST USING INDEX
SEARCH KEY IS K-REGION, K-BRANCHE          USING INDEX   * keine Replikate*

FETCH KUNDE USING K-BRANCHE ='ELEKTRO'
FETCH  KUNDE USING K-REGION = 7
```

Beispiel 2

Aufgabe sei, die 'Großkunden' einer Region zu ermitteln. Dafür standen in UDS/SQL V2.0 unterschiedliche Möglichkeiten zur Verfügung, die entweder Datenreplikation erfordern oder nur teilweise über die Indextabelle abgearbeitet werden können. Für die Gegenüberstellung werden nur die beiden Varianten ohne Datenreplikation betrachtet:

UDS/SQL V2.0:

```
SEARCH KEY IS K-VOLUMEN          USING INDEX

FETCH KUNDE USING K-VOLUMEN >=10000 AND K-REGION = 8
```

oder mit zwei Search Keys

```
SEARCH KEY IS K-VOLUMEN          USING INDEX
SEARCH KEY IS K-REGION           USING INDEX

FETCH KUNDE USING K-VOLUMEN >=10000 AND K-REGION = 8
```

Im ersten Fall kann nur K-VOLUMEN über die Indextabelle ausgewertet werden; die so gebildete Obermenge wird sequenziell nach K-REGION = 8 durchsucht.
Im zweiten Fall werden bezüglich K-VOLUMEN und K-REGION Zwischentreffermengen gebildet und geschnitten. Diese Lösung hat für Update-Funktionen Nachteile, weil eine zusätzliche Search-Key-Tabelle gepflegt werden muss.

ab UDS/SQL V2.2:

```
SEARCH KEY IS K-VOLUMEN, K-REGION         USING INDEX

FETCH KUNDE USING K-VOLUMEN >=10000 AND K-REGION = 8
```

Verwendung von Compound Keys in Suchausdrücken mit den boole'schen Operatoren AND und OR

Compound Keys zeigen ihre volle Wirkung nur innerhalb von Suchausdrücken, die entweder nur den boole'schen Operator AND oder nur OR enthalten.
Treten beide Operatoren gemischt in einem Suchausdruck auf, werden jeweils die AND-Bestandteile ('AND-Gruppe') und die OR-Bestandteile (OR-Gruppe) getrennt ausgewertet.

Allen folgenden Beispielen liegen die hier definierten Search Keys zu Grunde:

```
SEARCH KEY IS K-BRANCHE, K-REGION, K-PLZREST USING INDEX
SEARCH KEY IS K-REGION, K-BRANCHE            USING INDEX
SEARCH KEY IS K-VOLUMEN, K-REGION            USING INDEX

FETCH KUNDE USING K-BRANCHE = 'ELEKTRO'
                     AND (K-REGION = 2 OR K-REGION = 8)
```

K-BRANCHE und K-REGION können nicht im gleichen Durchgang ausgewertet werden.

Die Zuordnung der Suchkriterien zu Indextabellen wird nach den unten beschriebenen Regeln vorgenommen, wenn das Suchkriterium aus mehr als einer Indextabelle als Schlüsselfeld ausgewählt werden kann.
Zu diesem Zweck unterscheidet man zwischen "Hauptschlüsselfeldern" und "Nebenschlüsselfeldern". Hauptschlüsselfelder bilden innerhalb eines Compound Key das erste Schlüsselfeld, Nebenschlüsselfelder treten als zweites bis n-tes Schlüsselfeld auf. Dabei kann es vorkommen, dass ein Suchkriterium Hauptschlüsselfeld in der einen Indextabelle und Nebenschlüsselfeld in der anderen ist; z.B. sind K-BRANCHE und K-REGION in den ersten beiden Search Keys aus dem Beispiel jeweils kreuzweise einmal Hauptschlüsselfeld und einmal Nebenschlüsselfeld. Der Fall, dass ein Schlüsselfeld Hauptschlüsselfeld in mehreren Indextabellen ist wird nicht behandelt, die Auswahl der Indextabelle ist dann unbestimmt.

AND-Gruppen

Die Suchbedingungen werden in der vorgegebenen Reihenfolge abgearbeitet.
Nach Auffinden des ersten Hauptschlüsselfeldes wird zunächst geprüft, ob es dazu passende Nebenschlüsselfelder im Suchausdruck gibt (d.h. zur gleichen Indextabelle gehörige Schlüsselfelder), bevor weitere Hauptschlüsselfelder gesucht werden.

Beispiele

```
FETCH KUNDE USING K-BRANCHE = 'ELEKTRO'  AND  K-REGION = 2
```

K-BRANCHE ist das zuerst auftretende Hauptschlüsselfeld. Obwohl K-REGION auch ein Hauptschlüsselfeld ist (in einer anderen Indextabelle), wird es als Nebenschlüsselfeld K-BRANCHE beigeordnet.

```
FETCH KUNDE USING K-REGION > 7  AND  K-BRANCHE = 'ELEKTRO'
                                     AND  K-VOLUMEN > 100000
```

K-REGION ist das zuerst auftretende Hauptschlüsselfeld. Obwohl K-BRANCHE auch ein Hauptschlüsselfeld ist (in einer anderen Indextabelle), wird es als Nebenschlüsselfeld K-REGION beigeordnet.
K-VOLUMEN wird als zweites Hauptschlüsselfeld ausgewählt.

```
FETCH KUNDE USING K-BRANCHE NOT= 'ELEKTRO'  AND  K-REGION = 2
```

Eigentlich ist K-BRANCHE das zuerst auftretende Hauptschlüsselfeld. Da aber "NOT=" eine sehr unselektive Bedingung ist, wird es nicht als Hauptschlüsselfeld anerkannt, sondern stattdessen wird K-REGION im anderen Search Key ausgewählt und K-BRANCHE diesem als Nebenschlüsselfeld beigeordnet.

Ein Compound Calc Key kann nur innerhalb einer AND-Gruppe ausgewertet werden, und auch nur dann, wenn das Haupt- und sämtliche Nebenschlüsselfelder (egal in welcher Reihenfolge) mit "="-Operator ohne Maske aufgeführt sind.

Nach der Zuordnung der Suchbedingungen zu Index- oder Calc-Tabellen wird entschieden, in welcher Reihenfolge die Tabellen ausgewertet werden:

1. Suchbedingung als Single Key oder Datengruppen-Key mit "="-Operator
2. Suchbedingung als Single Key oder Datengruppen-Key mit "<" - oder ">" - Operator, oder als Hauptschlüsselfeld in einem Compound Key in Verbindung mit einem "=" -, "<" - oder ">" - Operator zusammen mit den zugehörigen Nebenschlüsselfeldern
3. alle sonstigen Suchbedingungen

OR-Gruppen

Falls eine OR-Gruppe vorliegt, werden die Suchbedingungen in der Reihenfolge des Auftretens abgearbeitet. Dabei werden nur die Hauptschlüsselfelder für einen gezielten Tabellenzugriff ausgewählt. Nebenschlüsselfelder werden zwar ebenfalls via Tabellenzugriff ausgewertet, aber ohne Eingrenzung des Suchbereichs. Beispiel:

```
FETCH KUNDE USING K-BRANCHE = 'ELEKTRO'  OR  K-REGION = 2
```

Beide Felder sind Hauptschlüsselfelder zweier verschiedener Indextabellen und werden als solche ausgewählt.

Einschränkungen

In folgenden Fällen kann UDS/SQL den Direktzugriff trotz Vorhandenseins einer Tabelle nicht oder nur teilweise ausnutzen:

– Die Schlüsselbedingung ist durch "OR" mit einer Nicht-Schlüsselbedingung verknüpft.

– Es handelt sich um einen CALC-Schlüssel und eine der folgenden Bedingungen ist erfüllt:

  – *satzelementname-6* ist maskiert
  – es handelt sich um *satzelementname-7* (NEXT)
  – *bedingung-1* oder Vergleichsoperator ist negiert (NOT)
  – Vergleichsoperator ist nicht „EQUAL“.

– Der Schlüssel gehört zu einer Liste und ist maskiert, die Maske beginnt nicht mit HIGH VALUE (LOW VALUE-Maske).

– Der Schlüssel ist ein DESCENDING KEY.

– Es handelt sich um einen Schlüssel in *komplex-2* (NEXT) und eine der folgenden Bedingungen ist erfüllt:

  – *komplex-2* besteht aus mehr als einer Bedingung
  – der Suchausdruck besteht aus *komplex-1* und *komplex-2*

- Sequenzielle Suche mit Single Sweep

  DIRECT-CALC-Sätze werden in der Reihenfolge ihrer physischen Position gelesen, d.h. mit Single Sweep.

  Dieser Single Sweep wird nur bei FIND3/7 auf Satzartebene angewendet, während die Suche bei FIND FIRST und FIND NEXT entlang der DBTT erfolgt.

Behandlung von dynamischen Sets bei einem Statuscode ≠ 000

Findet ein FIND/FETCH-7 USING *suchausdruck* keinen Treffer (Statuscode = 024), so sind der implizite dynamische Set und ein evtl. angegebener expliziter dynamischer Set anschließend leer.
Scheitert ein FIND/FETCH-7 USING *suchausdruck* (Statuscode ≠ 000 und ≠ 024), so werden der implizite dynamische Set und ein evtl. angegebener expliziter dynamischer Set auf den Stand vor dem FIND/FETCH-7 gesetzt.

Beispiel 1

```
      FETCH-7-SATZELEMENTNAMEN.
           MOVE 1 TO BEST-MONAT.
           MOVE 82 TO BEST-JAHR.
           FETCH BESTELLUNG USING BEST-MONAT, BEST-JAHR; DISPLAY
                 "BEST-NR = " BEST-NR "     DATUM = " BEST-TAG
                 BEST-MONAT BEST-JAHR UPON TERMINAL.
           PERFORM DUPLIKATE-LESEN-BEST UNTIL DATABASE-STATUS = 04021.
           DISPLAY "KEIN WEITERES DUPLIKAT VORHANDEN" UPON TERMINAL.
           GO TO FETCH-7-SUCHAUSDRUCK.
       DUPLIKATE-LESEN-BEST.
           FETCH DUPLICATE WITHIN BESTELLUNG USING BEST-MONAT, BEST-JAHR;
           IF DATABASE-STATUS IS NOT = 04021 DISPLAY "BEST-NR = "
           BEST-NR "     DATUM = " BEST-TAG BEST-MONAT BEST-JAHR
           UPON TERMINAL.
*****************************************************************
(OUT)    BEST-NR = 0003     DATUM = 150182
(OUT)    BEST-NR = 7995     DATUM = 100182
(OUT)    BEST-NR = 8540     DATUM = 030182
(OUT)    BEST-NR = 7854     DATUM = 030182
(OUT)    KEIN WEITERES DUPLIKAT VORHANDEN
```

Die erste FETCH-Anweisung wählt einen Satz aus, der das in der UWA vorgegebene Monats- und Jahresdatum aufweist.
Durch eine anschließende Folge von FETCH DUPLICATE werden alle weiteren Bestellungen ausgegeben, die dasselbe Monats- und Jahresdatum besitzen.

Bei FETCH DUPLICATE ist die Angabe von USING notwendig.

Beispiel 2

```
 WORKING-STORAGE SECTION.
 01  ENDE-BEDINGUNG                 PIC XX.
     88 ENDE                        VALUE IS "JA".
*
 FETCH-7-SUCHAUSDRUCK.
     FIND ARTIKELBESCHR USING BEZEICHNUNG OF ARTIKELBESCHR IS
          = "SOMMERKLEID MIT JACKE" RESULT IN ERGEBNISSET.
     FETCH ARTIKEL WITHIN BESTELLANGABEN USING PREIS IS < 300
          AND GROESSE IS = 36 AND AKT-BESTAND IS > MIN-BESTAND;
     PERFORM DUPLIKATE-LESEN-ARTIKEL UNTIL DATABASE-STATUS = 04021.
     PERFORM DUPLIKATE-LESEN-ARTIKELBESCHR UNTIL ENDE.
     GO TO MODIFY-SATZELEMENT.
 DUPLIKATE-LESEN-ARTIKELBESCHR.
     FIND NEXT RECORD WITHIN ERGEBNISSET; IF DATABASE-STATUS
          = 04021 MOVE "JA" TO ENDE-BEDINGUNG ELSE PERFORM
          DUPLIKATE-LESEN-ARTIKEL UNTIL DATABASE-STATUS = 04021.
 DUPLIKATE-LESEN-ARTIKEL.
     FETCH DUPLICATE WITHIN BESTELLANGABEN; IF DATABASE-STATUS
          IS NOT = 04021 DISPLAY ARTIKEL UPON TERMINAL.
```

Dies ist ein Beispiel für eine komplexe Suchfrage: Für die Suche nach Artikelsätzen werden zuerst Felder der Satzart ARTIKELBESCHR benutzt und die Treffersätze in einen dynamischen Set zwischengespeichert. Anschließend wird jede Set-Occurrence eines Treffersatzes nach Feldinhalten der Membersatzart ARTIKEL durchsucht.

FINISH

Die FINISH-Anweisung beendet eine Transaktion und gibt reservierte Realms und Seiten frei.

```
FINISH[ WITH CANCEL]
```

WITH CANCEL
: bei dieser Angabe macht der DBH alle Änderungen dieser Transaktion mit Rollback rückgängig.

Ein FINISH (ohne CANCEL) in openUTM-Programmen wird bis zum PEND verzögert.

FREE

Die FREE-Anweisung beendet den erweiterten Satzschutz durch KEEP für einen oder mehrere Sätze:

```
FREE[ ALL]
```

ALL
: Mit dieser Angabe geben Sie alle Sätze, die mit KEEP zusätzlich geschützt waren, für andere Transaktionen frei, die normale CRU-Sperre bleibt erhalten.
Ohne diese Angabe heben Sie nur den zusätzlichen Satzschutz des CRU auf.

GET

Die GET-Anweisung transportiert den CRU ganz oder teilweise in die UWA.

```
GET[ {satzname              }]
     {satzelementname,...   }
```

satzname
: muss die Satzart des CRU bezeichnen. UDS/SQL transportiert dann den gesamten CRU in die UWA.

satzelementname,...
: müssen aus der Satzart des CRU stammen. Sie treffen damit eine Auswahl von Satzelementen des CRU, deren Inhalt UDS/SQL in die UWA transportieren soll.

Wenn Sie weder *satzname* noch *satzelementname,...* angeben, bringt UDS/SQL den gesamten CRU in die UWA.

IF

Die IF-Anweisung prüft im Programm Set-Mitgliedschaften. Es gibt zwei Formate:

Format 1:

```
IF[ NOT][ setname] {OWNER  } {anweisung-1 }[ ELSE {anweisung-2 }].
                   {MEMBER }{             }      {             }
                   {TENANT } {NEXT SENTENCE}      {NEXT SENTENCE}
```

setname
darf kein dynamischer Set sein; beschränkt die Prüfung auf den angegebenen Set. Geben Sie *setname* nicht an, werden alle Sets des Subschemas geprüft, in denen die Satzart des CRU Owner- oder Membersatzart ist.

OWNER
prüft, ob der CRU Ownersatz einer nicht leeren Set-Occurrence ist,

MEMBER
prüft, ob der CRU Membersatz in einer Set-Occurrence ist,

TENANT
prüft, ob der CRU Ownersatz einer nicht leeren Set-Occurrence oder Membersatz in einer Set-Occurrence ist.

Format 2:

```
IF setname IS[ NOT] EMPTY {anweisung-1 }[ ELSE {anweisung-2 }].
                          {             }      {             }
                          {NEXT SENTENCE}      {NEXT SENTENCE}
```

setname
wählt über den CRS die Set-Occurrence aus und prüft, ob diese leer ist. *setname* darf kein dynamischer Set sein.
Wenn der CRS ein gelöschter oder ausgehängter Membersatz ist, wird der Statuscode 031 abgesetzt.

anweisung-1 bis *anweisung-2*
können sowohl beliebige COBOL-Anweisungen als auch DML-Anweisungen sein.

Tritt ein Datenbanksonderzustand ein, wird „UNWAHR“ erkannt und nach *anweisung-2* bzw. NEXT SENTENCE verzweigt.

KEEP

Die KEEP-Anweisung schützt den CRU vor dem verändernden Zugriff durch andere Verarbeitungsketten und Transaktionen, auch wenn ein anderer Satz zum CRU geworden ist.

```
KEEP
```

Die KEEP-Anweisung bedeutet in einer Transaktion, dass der CRU solange für andere Transaktionen gesperrt ist, bis eine FREE- oder FINISH-Anweisung gegeben wurde. Gesperrt wird immer die Seite, in der der CRU steht. In einer RETRIEVAL-Transaktion wird diese Seite gegen Änderungen gesperrt, in einer UPDATE-Transaktion gegen jeden Zugriff.

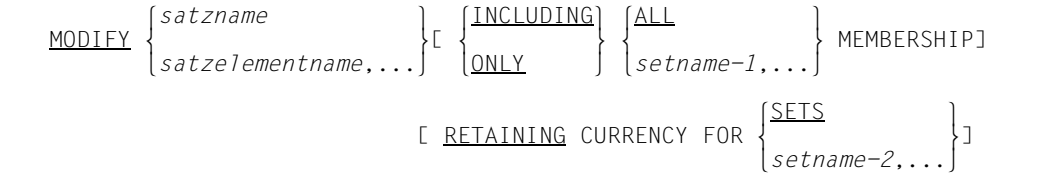
MODIFY

Die MODIFY-Anweisung kann

– alle oder eine Auswahl von Feldinhalten des CRU durch Werte aus der UWA ersetzen

– den CRU innerhalb eines Set in eine andere Set-Occurrence umhängen und

– den CRU zum CRS aller Sets machen, in denen Sie seine Position ändern. CRA und CRR werden nicht verändert.



satzname
: muss die Satzart des CRU bezeichnen.
Wenn Sie nicht ONLY angeben, überträgt UDS/SQL alle Feldinhalte der zur Satzart gehörenden Felder aus der UWA auf den CRU. Die Felder müssen Sie zuvor in der UWA mit den gewünschten Werten versorgen.

satzelementname,...
: müssen Satzelemente aus der Satzart des CRU, die kein Feld variabler Länge enthalten darf, bezeichnen. Sie zählen damit die Satzelemente auf, deren Inhalte aus der UWA in den CRU zu transportieren sind. Die Satzelemente müssen Sie zuvor mit den gewünschten Werten versorgen.
Wenn Sie Schlüsselwerte ändern, aktualisiert UDS/SQL automatisch alle zugehörigen Tabellen. Insbesondere kann sich dadurch die Reihenfolge von Sätzen innerhalb einer Set-Occurrence ändern. Den Database-Key-Wert eines Satzes können Sie nicht ändern.

INCLUDING
: ist nur erlaubt, wenn der CRU in mindestens einer Set-Occurrence enthalten ist. INCLUDING ändert die Feldinhalte des CRU und hängt den CRU in andere Set-Occurrences um. Welche Sets davon betroffen sind, bestimmen sie mit ALL oder *setname*,.... In diesen Sets müssen Sie zuvor die Set-Occurrences und ggf. auch die Positionen innerhalb der Set-Occurrences auswählen, in die der CRU umgehängt werden soll. Die Auswahl kann genau wie bei FIND und STORE durch die SET OCCURRENCE SELECTION THRU LOCATION MODE OF OWNER definiert sein (siehe auch FIND-7 auf Seite 164 sowie STORE (Set-Occurrence auswählen) auf Seite 192 und STORE (Einfügungsort innerhalb einer Set-Occurrence bestimmen) auf Seite 193).

ONLY darf nicht zusammen mit *satzelementname*,... verwendet werden und ist nur erlaubt, wenn der CRU in mindestens einer Set-Occurrence enthalten ist.
ONLY lässt die Feldinhalte des CRU unverändert und hängt den CRU in andere Set-Occurrences der betroffenen Sets um. Die Auswahl der Set-Occurrence treffen Sie genau wie bei INCLUDING.

ALL hängt den CRU in allen Sets um, in denen er einer Set-Occurrence angehört. Dynamische Sets sind ausgeschlossen.

setname-1,...
muss einen Set bezeichnen, dem der CRU einer Set-Occurrence angehört. Sie zählen damit die Sets auf, in denen der CRU einem neuen Ownersatz zugeordnet werden soll. Die Angabe eines dynamischen Set ist wirkungslos.

RETAINING
Sie können damit das Aktualisieren der CURRENCY-Tabelle gezielt dort unterdrücken, wo Sie den Database-Key-Wert des vorangegangenen Satzes bewahren wollen.

RETAINING gilt automatisch für alle Sets, die nicht Objekt der MODIFY-Anweisung sind, die Angabe eines dynamischen Set ist daher wirkungslos.

SETS unterdrückt das Aktualisieren aller CRS.

setname-2,...
unterdrückt das Aktualisieren der CRS der angegebenen Sets.

Beispiel 1

```
       MODIFY-SATZELEMENT.
           FETCH FIRST LIEFERANT; DISPLAY "LIEFER-NAME = " LIEFER-NAME
                "HAUSNR = " LIEFER-HAUSNR UPON TERMINAL.
           MOVE 10 TO LIEFER-HAUSNR.
           MODIFY LIEFER-HAUSNR.
******************************************************************
(OUT)    LIEFER-NAME = MONA MODEHAUS                HAUSNR = 1
```

Die Ausgabe zeigt einen Lieferanten mit der zu ändernden Hausnummer. MODIFY ändert hier die Hausnummer 1 in Hausnummer 10.

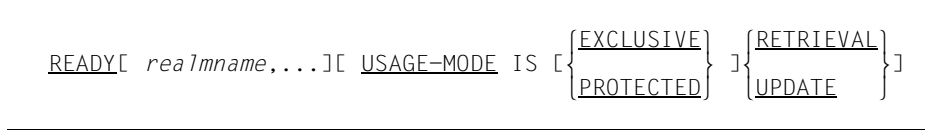
Beispiel 2

```
       MODIFY-OWNERZUORDNUNG.
           FETCH OWNER WITHIN ABGEGEBENE-BEST; DISPLAY "LIEFER-NAME = "
                LIEFER-NAME "ADRESSE = " LIEFER-STRASSE " "
                LIEFER-HAUSNR "   " LIEFER-STADT UPON TERMINAL.
           MOVE 2 TO LIEFER-NR.
           MOVE "AUGUSTINERBRAEU" TO LIEFER-NAME.
           FIND ANY LIEFERANT.
           FIND CURRENT BESTELLUNG RETAINING CURRENCY FOR MULTIPLE.
           MODIFY BESTELLUNG ONLY ABGEGEBENE-BEST MEMBERSHIP.
           FETCH OWNER WITHIN ABGEGEBENE-BEST; DISPLAY "LIEFER-NAME = "
                LIEFER-NAME "ADRESSE = " LIEFER-STRASSE " "
                LIEFER-HAUSNR "   " LIEFER-STADT UPON TERMINAL.
******************************************************************
(OUT)    LIEFER-NAME = MONA MODEHAUS     ADRESSE = AUGUSTENSTRASSE 10
                                                  KARLSRUHE 1
(OUT)    LIEFER-NAME = AUGUSTINERBRAEU   ADRESSE = MARSSTRASSE 77
                                                  MUENCHEN 2
```

Eine Bestellung, die irrtümlich dem Lieferanten MONA MODEHAUS zugeordnet wurde, wechselt in die Set-Occurrence des Lieferanten AUGUSTINERBRAEU.

READY

Die READY-Anweisung eröffnet eine Transaktion oder eine Verarbeitungskette und bereitet einen oder mehrere Realms für die Verarbeitung vor.



realmname,...
: die angegebenen Realms des Subschemas werden eröffnet.
Geben Sie *realmname*,... nicht an, dann werden alle zum Subschema gehörenden Realms eröffnet, auch der Temporäre Realm.

USAGE-MODE
: hiermit definieren Sie eine Benutzungsart für die betroffenen Realms und geben dadurch die Art der erlaubten Zugriffe an

EXCLUSIVE
: keine weitere Verarbeitungskette darf gleichzeitig mit den Realms arbeiten

PROTECTED
: andere Verarbeitungsketten dürfen gleichzeitig nicht ändern

RETRIEVAL
: Daten wiedergewinnen

UPDATE
: Daten wiedergewinnen und ändern

Geben Sie USAGE-MODE nicht an, wird USAGE-MODE RETRIEVAL angenommen.

SET

Die SET-Anweisung überträgt den Inhalt eines Database-Key-Feldes in ein oder mehrere Database-Key-Felder.

SET *feldname-1*,... TO *feldname-2*

---

feldname-1,...
: muss mit USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG definiert sein.

feldname-2
: muss mit USAGE IS DATABASE-KEY oder USAGE IS DATABASE-KEY-LONG definiert sein.

Damit Database-Key-Werte mit einer REC-REF > 254 und/oder einer RSQ > 2^{24}-1 korrekt übertragen werden können, müssen *feldname-2* und alle *feldname-1,...* mit USAGE IS DATABASE-KEY-LONG definiert sein.

Bei der Übertragung von Database-Key-Werten sind folgende Fälle zu unterscheiden:

- *feldname-2* und alle Felder *feldname-1,...* sind vom Typ USAGE IS DATABASE-KEY.
  In diesem Fall überträgt die SET-Anweisung den Wert von *feldname-2* jeweils eins zu eins in alle Felder *feldname-1,...*.
  Bei korrekter Übertragung bleibt der DATABASE-STATUS unverändert.
- *feldname-2* ist vom Typ USAGE IS DATABASE-KEY; mindestens eins der Felder *feldname-1,...* ist vom Typ USAGE IS DATABASE-KEY-LONG.
  Bei der Übertragung von DATABASE-KEY nach DATABASE-KEY-LONG verfährt die SET-Anweisung wie folgt:
  - Die Satzartnummer (REC-REF) der Länge 1 byte wird rechtsbündig in den entsprechenden 2 byte langen Bereich des Empfangsfeldes übertragen.
  - Die Satzfolgenummer (RSQ) der Länge 3 byte wird rechtsbündig in den entsprechenden 4 byte langen Bereich des Empfangsfeldes übertragen.

  Bei korrekter Übertragung wird der DATABASE-STATUS 00000 ausgegeben.
- *feldname-2* und alle Felder *feldname-1,...* sind vom Typ USAGE IS DATABASE-KEY-LONG.
  In diesem Fall überträgt die SET-Anweisung den Wert von *feldname-2* jeweils eins zu eins in alle Felder *feldname-1,...*.
  Bei korrekter Übertragung wird der DATABASE-STATUS 00000 ausgegeben.

- *feldname-2* ist vom Typ USAGE IS DATABASE-KEY-LONG; mindestens eins der Felder *feldname-1,...* ist vom Typ USAGE IS DATABASE-KEY.
  Bei der Übertragung von DATABASE-KEY-LONG nach DATABASE-KEY verfährt die SET-Anweisung wie folgt:

  – Die Satzartnummer (REC-REF, ursprüngliche Länge 2 byte) wird in den entsprechenden 1 byte langen Bereich des Empfangsfeldes übertragen.

  – Die Satzfolgenummer (RSQ, ursprüngliche Länge 4 byte) wird in den entsprechenden 3 byte langen Bereich des Empfangsfeldes übertragen.

  Wird zur Laufzeit festgestellt, dass bei der Übertragung Datenverlust (Abschneiden führender Stellen) wegen zu großer Satzartnummer und/oder Satzfolgenummer auftritt, überträgt die SET-Anweisung den Database-Key-Wert 0 in das betreffende Empfangsfeld. In diesem Fall meldet UDS/SQL den DATABASE-STATUS 00102. Die SET-Anweisung wird jedoch nicht abgebrochen, d.h. die restlichen Übertragungen werden durchgeführt.
  Bei korrekter Übertragung des Database-Key-Wertes in alle Empfangsfelder wird der DATABASE-STATUS 00000 ausgegeben.

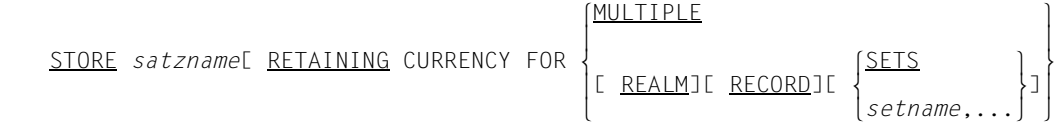
STORE

Die STORE-Anweisung

- überträgt einen Satz aus der UWA als neuen Satz in die Datenbank,
- fügt den neuen Satz in alle Sets ein, für die seine Satzart im Schema als AUTOMATIC Member definiert ist,
- richtet eine neue Set-Occurrence mit den zugehörigen Tabellen für jeden Set ein, für den die Satzart des neuen Satzes im Schema als Ownersatzart definiert ist und für den mit der SSL die Set-Occurrence-Population größer als 0 definiert ist und
- macht den neuen Satz zum aktuellen Satz in allen betroffenen Spalten der Currency-Tabelle, in denen Sie das Aktualisieren nicht mit RETAINING unterdrücken. Betroffen sind die Spalten der zugehörigen Satzart, aller Sets, in denen der neue Satz Member oder Owner ist, und des Realm, in den der Satz gespeichert wird.

Voraussetzung ist, dass im Subschema mindestens vorhanden sind:

- alle in der LOCATION MODE-Klausel der zugehörigen Satzart genannten Felder,
- alle Sets, für die die Satzart im Schema als AUTOMATIC Member erklärt ist und
- alle Felder, die zur eindeutigen Auswahl der Ownersätze dieser Sets gebraucht werden, wenn die Auswahlmethode mit SELECTION THRU LOCATION MODE OF OWNER definiert ist.



satzname
bezeichnet den einzuspeichernden Satz. Dessen Felder müssen Sie zuvor in der UWA mit den gewünschten Werten versorgen. Felder der Satzart, die nicht im Subschema enthalten sind, füllt UDS/SQL beim Einspeichern automatisch mit binären Nullen.

RETAINING
müssen Sie entweder mit MULTIPLE oder mindestens einer der Angaben REALM, RECORD, SETS oder *setname*,... verwenden.

RETAINING gilt automatisch für alle Sets, die nicht Objekt der STORE-Anweisung sind, die Angabe eines dynamischen Sets ist daher wirkungslos.

Realm auswählen

Wenn im Schema mehrere Realms für die Aufnahme von Sätzen dieser Satzart vorgesehen sind, so muss ein Realm bestimmt werden:

1. Der Realm ist eindeutig bestimmt, wenn die Satzart Member in einem Set ist und im Schema PLACEMENT OPTIMIZATION für die Satzart oder MODE IS LIST für den Set definiert ist.
   In diesem Fall speichert UDS/SQL den Satz
   – in den Realm des zugehörigen Ownersatzes, oder
   – in den Realm der DETACHED WITHIN-Klausel im Set-Eintrag der SSL oder
   – im ersten Realm der DDL WITHIN-Klausel der Membersatzart,
   auch wenn Sie entsprechend der folgenden Vorgehensweise einen anderen Realm auswählen.
   Beim Abspeichern eines Satzes, der Member einer verteilbaren Liste ist, wird im DBH eine benötigte freie Seite in dem bevorzugten Realm ((Preferred-Realm) gesucht. Die Angabe eines Realms im AREA-ID-Feld ist in diesem Fall bedeutungslos.
2. In den übrigen Fällen müssen Sie den Realm selbst bestimmen, indem Sie den gewünschten Realm-Namen in das Feld übertragen, das in der WITHIN-Klausel des Schemas als AREA-ID-Feld für diese Satzart definiert ist (siehe Handbuch „Entwerfen und Definieren“, WITHIN-Klausel).
   UDS/SQL ignoriert diese Angabe, wenn der Realm nach 1) bestimmt ist.

Database-Key-Wert zuordnen

Jedem zu speichernden Satz wird vom DBH ein Database-Key-Wert zugeordnet. Nach dem Canceln einer speichernden Transaktion sind die vergebenen Database-Key-Werte zwar im Allgemeinen wieder frei, werden jedoch beim weiteren Speichern zunächst nicht neu vergeben. Erst wenn das Ende der DBTT erreicht ist, wird abhängig von einer aktivierten Online-DBTT-Erweiterung ggf. vom Beginn der DBTT an neu vergeben.

Ist in der Schema-DDL die Vergabe von Database-Key-Werten durch den Anwender vorgesehen, so ist dort ein Feld für die Aufnahme von Database-Key-Werten definiert (siehe Handbuch „Entwerfen und Definieren“, LOCATION MODE-Klausel). Dieses Feld müssen Sie vor dem Einspeichern eines Satzes mit dem gewünschten Wert versorgen. Auf diese Weise können Sie den Database-Key-Wert jedes Satzes als Informationsträger nutzen (z.B. als Kundennummer); mit FIND-1 können Sie sehr schnell direkt über den Database-Key-Wert auf einen Satz zugreifen.

Mit einem von Ihnen vergebenen Database-Key-Wert verfährt UDS/SQL wie folgt:

- UDS/SQL ermittelt in jedem Fall die zur Satzart *satzname* gehörende Satzartnummer (REC-REF) selbstständig; die von Ihnen vergebene Satzartnummer wird ignoriert. Dies bedeutet, dass UDS/SQL auch bei Angabe einer falschen Satzartnummer stets den gewünschten (korrekten) Database-Key-Wert vergibt.

- In folgenden Fällen ignoriert UDS/SQL die von Ihnen vergebene Satzfolgenummer (RSQ) und vergibt selbstständig eine verfügbare Satzfolgenummer:
  - Die von Ihnen vergebene Satzfolgenummer hat den Wert 0.
  - Die von Ihnen vergebene Satzfolgenummer ist größer als die aktuelle Anzahl Zeilen in der Database Key Translation Table (DBTT) für die Satzart *satzname* (siehe Handbuch „Entwerfen und Definieren“). Die Vergabe einer größeren Satzfolgenummer führt also nicht zu einer Online-DBTT-Erweiterung.
  - Für die Satzart *satzname* gibt es bereits einen Satz mit der von Ihnen vergebenen Satzfolgenummer.

Wenn die Satzart *satzname* im Schema nicht mit LOCATION MODE IS DIRECT bzw. LOCATION MODE IS DIRECT-LONG definiert ist, werden die Database-Key-Werte für die Sätze der Satzart *satzname* von UDS/SQL vergeben.

Mit ACCEPT können Sie prüfen, ob der tatsächliche Database-Key-Wert mit Ihrer Angabe übereinstimmt (siehe Beispiele zu FETCH-1 ab Seite 152 und zu ACCEPT ab Seite 140).

Set-Occurrence auswählen

In jedem Set, in dem die Satzart *satzname* AUTOMATIC Member ist, wird der eingespeicherte Satz automatisch in eine Set-Occurrence eingehängt. Deshalb müssen Sie vorher die Ownersätze des einzuspeichernden Satzes auswählen. Sie müssen dazu die Auswahlmethoden benutzen, die im Schema für diese Sets festgelegt sind (siehe Handbuch „Entwerfen und Definieren“, SET OCCURRENCE SELECTION-Klausel). Entsprechend der Schema-Definition ist die Set-Occurrence entweder durch den CRS bestimmt oder Sie müssen den Database-Key- bzw. CALC-Key-Wert des zugehörigen Ownersatzes in die UWA über-

tragen. UDS/SQL wählt die Set-Occurrence dann automatisch aus, indem es über den Database Key bzw. das Hashverfahren auf den Ownersatz zugreift. Die folgende Übersicht fasst noch einmal zusammen, wie Sie eine Set-Occurrence auswählen.

<table>
<tr><td rowspan="3">Schema-Definition</td><td colspan="3">SET OCCURRENCE SELECTION IS THRU</td></tr>
<tr><td>CURRENT OF SET</td><td colspan="2">LOCATION MODE OF OWNER[ALIAS FOR { feldname | bezeichner-1 } IS bezeichner-2]...

LOCATION MODE IS</td></tr>
<tr><td>{ DIRECT | DIRECT-LONG } { feldname | bezeichner-1 }]</td><td>CALC USING feldname,...</td></tr>
<tr><td>Sie bestimmen die Set-Occurrence durch:</td><td>CRS</td><td>Database-Key-Wert im Feld feldname bzw. bezeichner in UWA bzw. in den in der ALIAS-Klausel genannten Feldern</td><td>CALC-Key-Wert in den Feldern feldname,... bzw. bezeichner-2 in UWA bzw. in den in der ALIAS-Klausel genannten Feldern</td></tr>
</table>

Tabelle 21: Auswahl einer Set-Occurrence

Einfügungsort innerhalb der Set-Occurrence bestimmen

In einer ausgewählten Set-Occurrence bekommt der eingespeicherte Satz den Platz, der der im Schema für den Set definierten Reihenfolge entspricht (siehe Handbuch „Entwerfen und Definieren“, ORDER-Klausel). Die nachfolgende Tabelle enthält die Fälle, in denen der Einfügungsort von Ihren Angaben abhängt.

<table>
<tr><td rowspan="2">Schema-Definition</td><td colspan="2">ORDER IS</td></tr>
<tr><td>NEXT/PRIOR

SET OCCURRENCE SELECTION IS THRU CURRENT OF SET</td><td>SORTED[INDEXED] BY DEFINED KEYS

{ ASCENDING | DESCENDING } KEY IS feldname,...</td></tr>
<tr><td>Sie bestimmen den Einfügungsort durch:</td><td>CRS</td><td>Schlüsselwert in Feldern feldname,... im einzuspeichernden Satz</td></tr>
</table>

Tabelle 22: Einfügungsort in einer Set-Occurrence bestimmen

Im Falle SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER hat der CRS keine Bedeutung, der Einfügungsort bei ORDER IS NEXT bzw. PRIOR liegt am Anfang bzw. Ende der Set-Occurrence wie bei ORDER IS FIRST bzw. LAST.

Beispiel

```
STORE-ARTIKEL.
   MOVE "KLEIDUNG" TO RLMAUSWAHL-4.
   MOVE 4 TO ART-NR OF ARTIKELBESCHR.
   FIND ARTIKELBESCHR USING ART-NR OF ARTIKELBESCHR.
   MOVE "MONA MODEHAUS" TO LIEFER-NAME.
   FIND LIEFERANT USING LIEFER-NAME.
   MOVE 8 TO FARB-NR OF ARTIKEL.
   MOVE 36 TO GROESSE.
   MOVE 3721 TO ART-NR-LIEFER.
   MOVE 5 TO FARB-NR-LIEFER.
   MOVE 285 TO PREIS.
   STORE ARTIKEL.
```

Der Artikel mit Farb-Nr 8, Größe 36 usw. wird in den Realm KLEIDUNG eingespeichert und gleichzeitig in die Set-Occurrences von Artikelbeschreibung 4 und von Lieferant MONA MODEHAUS eingehängt.

USE

Die USE-Anweisung definiert Befehlsfolgen, die durchlaufen werden, wenn eine DML-Anweisung mit einem Datenbanksonderzustand endet.

```
USE FOR DATABASE-EXCEPTION[ ON { OTHER          }].
                               { literal-1,...  }
```

ON Bei dieser Angabe dürfen Sie mehrere USE-Anweisungen in Ihrem Programm verwenden. Die Literale müssen pro USE-Anweisung eindeutig sein.

Kommt eine USE-Anweisung ohne diese Angabe vor, muss sie die einzige im Programm sein.

Ohne diese Angabe oder bei ON OTHER wird die Befehlsfolge, die zu der USE-Anweisung gehört, bei jedem Datenbanksonderzustand ausgeführt.

OTHER
darf nur einmal in einem Programm verwendet werden;
muss angegeben werden, wenn nicht alle Datenbanksonderzustände durch *literal*,... angegeben wurden.

literal,...
Jedes *literal* muss einen Datenbanksonderzustand darstellen. Kein *literal* darf jedoch ’00000’ sein.

Die USE-Anweisung darf nur in den DECLARATIVES eines COBOL-Programmes angegeben werden.

Sie muss als erste Anweisung unmittelbar auf eine Kapitelüberschrift folgen. Dieses Kapitel muss die Befehlsfolgen enthalten, die auf Grund der USE-Anweisung durchlaufen werden sollen.

Die Befehlsfolgen dürfen weder die DECLARATIVES verlassen noch von den anderen Kapiteln des COBOL-Programms angesprochen werden. Eine solche Befehlsfolge darf die einer anderen USE-Anweisung nur mit PERFORM ansprechen.

Wenn ein Datenbanksonderzustand auftritt, werden die aktuellen Werte in den Sonderregistern abgespeichert. Treten bei einer DML-Anweisung mehrere Sonderzustände ein, so beziehen sich die Inhalte der Sonderregister auf den zuletzt festgestellten.

Beispiel

```
*************************************************************************
          .
          .
          .
        PROCEDURE DIVISION.
       *
        DECLARATIVES.
       *
        SCHUTZ SECTION.
            USE FOR DATABASE-EXCEPTION ON         "12901",
                                                  "12950",
                                                  "12951",
                                                  "12952",
                                                  "12953",
                                                  "12954",
                                                  "12955".
        M1. DISPLAY "*** DATABASE-STATUS = " DATABASE-STATUS " ***"
                                               UPON T.
           STOP RUN.
       *
        NOT-FOUND SECTION.
            USE FOR DATABASE-EXCEPTION ON         "04024".
        M2. MOVE HIGH-VALUE TO MERKER.
       *
        REST SECTION.
            USE FOR DATABASE-EXCEPTION            ON OTHER.
        M3. DISPLAY "****** FEHLER AUFGETRETEN! ******" UPON T.
            DISPLAY "DATABASE-STATUS = " DATABASE-STATUS UPON T.
            DISPLAY "SATZART         = " DATABASE-RECORD-NAME UPON T.
            DISPLAY "SET             = " DATABASE-SET-NAME UPON T.
            DISPLAY "REALM           = " DATABASE-REALM-NAME UPON T.
            FINISH WITH CANCEL.
            STOP RUN.
       *
      END DECLARATIVES.
       *
       *
       *
        PROGRAMM SECTION.
       *
        VORLAUF.
          .
          .
          .
*************************************************************************
```

8 Nachschlageteil der CALL-DML

Dieses Kapitel beschreibt die CALL-Schnittstelle der DML und die CALL-DML-Aufrufe einschließlich ihrer Parameterdefinitionen und Formate.

8.1 CALL-Schnittstelle

Die CALL-Schnittstelle gibt es in 2 Varianten:

– Die Namen der Datenbankobjekte müssen innerhalb der ersten 8 Stellen eindeutig sein, und zwar die Realm-, Set- und Satznamen innerhalb des jeweiligen Subschemas und die Feldnamen innerhalb des jeweiligen Satzes. Beim Datenbankentwurf muss bereits auf eine entsprechende Namenseindeutigkeit geachtet werden. Diese Variante wird im Folgenden mit (CALL8) bezeichnet.

– An die Namen der Datenbankobjekte ist keine so hohe Anforderung auf Eindeutigkeit gestellt. Diese Variante kann für beliebige Datenbanken verwendet werden. Sie wird im Folgenden mit (CALL30) bezeichnet.

Die CALL-Schnittstelle können Sie in allen Programmiersprachen verwenden, die sich beim Unterprogrammaufruf an die nachstehenden Registerkonventionen halten:

Register 15: Sprungziel

Register 14: Rücksprungadresse

Register 1: Adresse der Parameterleiste

Dies ist bei FORTRAN und COBOL der Fall.

Das Sprungziel ist für jeden DML-Aufruf gleich, die Sprungadresse heißt DML.

Die Adressen in der Parameterleiste (Register 1) folgen lückenlos aufeinander. Sie verweisen auf Felder im Programm, die die entsprechenden Parameter-Informationen enthalten. Bei COBOL und FORTRAN übernehmen die Compiler die Generierung der Parameterleiste, wenn der übliche CALL-Aufruf verwendet wird.

Beispiele zu den verschiedenen Programmiersprachen finden Sie auf Seite 289.

8.2 Parameterdefinitionen

Die CALL-DML kennt folgende Parameter:

1. Funktionsname: FCOD (function code)
2. Funktionswahl: FOPT (function option)
3. Zusatzwahl: SOPT (special option)
4. Benutzerinformation: UINF (user information)
5. Satzname: RECN (record name)
6. Setname: SETN (set name)
7. Realm-Name: RLMN (realm name)
8. Feldname: ITMN (item name)
9. Satzbereich: RECA (record area)
10. Spezialparameter-1: SPP1 (special parameter 1)
11. Spezialparameter-2: SPP2 (special parameter 2)
12. Spezialparameter-3: SPP3 (special parameter 3)

Die Parameter Funktionsname und Benutzerinformation werden bei jedem CALL-DML-Aufruf verwendet.
Welche der übrigen Parameter verwendet werden, ist abhängig von dem Funktionsnamen in Verbindung mit der Funktionswahl und der Zusatzwahl. Der Inhalt der ersten drei Parameter bestimmt, welche Parameterfelder der CALL-DML-Umsetzer liest oder beschreibt. Diese Parameter sind der Beschreibung der einzelnen Aufrufe und der Übersicht zu entnehmen.

Übergeben müssen Sie nur die Adressen der Parameterfelder. Die Länge der Parameterfelder ist in jedem Fall durch die CALL-DML-Vorschriften, durch das Subschema oder durch die Syntax des Feldinhalts (Namenslisten, Suchausdruck) festgelegt.

Jeder Parameter hat in der Parameterleiste des CALL-Aufrufs und in der zugehörigen Adressleiste einen festen Platz. Daher müssen Sie auch für die nicht verwendeten Parameter den Namen eines Feldes angeben, dessen Inhalt jedoch für die Ausführung des DML-Aufrufs belanglos ist. Der CALL-DML-Umsetzer wertet nur die Adressen der verwendeten Parameterfelder aus.

Diese Regelung gestattet es, mit demselben CALL alle möglichen DML-Anweisungen aufzurufen. Sie müssen vor der Ausführung des CALL-Befehls nur die Inhalte der verwendeten Parameterfelder entsprechend ändern.

Die Informationen, die in den Parameterfeldern enthalten sind, müssen in einem genau definierten Format abgelegt sein, damit sie sowohl der CALL-DML-Umsetzer als auch das Anwenderprogramm eindeutig interpretieren können.

Die Formate der einzelnen Parameterfelder sind soweit wie möglich vereinheitlicht, sodass ihre Anwendung sehr einfach ist. Abweichungen von den Standardformaten sind in den Beschreibungen der CALL-Aufrufe und in der Format-Tabelle genau erläutert.

Die Beschreibung der Parameter Zusatzwahl, Benutzerinformation sowie Spezialparameter-1 und Spezialparameter-2 finden Sie auf den Seiten 203 bis 210.
Syntax und Format des Suchausdrucks sind bei FIND7A/FTCH7A beschrieben.

8.2.1 Regeln

Die Parameterleiste brauchen Sie nicht in voller Länge anzugeben, wenn Sie am Ende der Leiste nur solche Parameter weglassen, die der CALL-Aufruf nicht verwendet.

- Der 1. Parameter ist immer der Funktionsname (FCOD).
- Die oben angegebene Reihenfolge muss eingehalten werden.
- Bis zum letzten verwendeten Parameter darf kein Parameterfeld ausgelassen werden.
- Wahlfreie Parameter ([ ]) zählen zu den verwendeten Parametern. Das zugehörige Parameterfeld muss mit Leerzeichen entsprechender Länge versorgt werden, wenn nichts anderes angegeben wird.
- Parameter, die in der Übersicht mit Bindestrich gekennzeichnet oder ohne Eintrag sind, zählen zu den nicht verwendeten Parametern.
- Der Inhalt nicht verwendeter Parameterfelder ist nicht relevant.
- Zu jedem Parameter gibt es für den Inhalt des Parameterfeldes Formatvorschriften, die von der Funktion abhängen können und die Sie einhalten müssen.
- Die Namen der Parameterfelder sind beliebig wählbar.
- Indizierte Felder können Sie in der CALL-DML nicht benennen, d.h. nicht einzeln auswählen. Deshalb müssen Sie bei indizierten Feldern immer mit dem vollständigen Subschemaformat des Satzes arbeiten.
- In den CALL-DML-Parametern treten an verschiedenen Stellen die Namen von Realms, Sets, Sätzen und Feldern auf.
  Bei der Variante (CALL8) sind diese Namen immer 8-stellig, bei der Variante (CALL30) immer 30-stellig anzugeben, gegebenenfalls mit Leerzeichen aufgefüllt.
  Die von Ihnen verwendete Variante wird über das Feld Benutzerinformationsende des Parameters UINF ausgewählt (siehe Seite 205):
  – Variante (CALL8): „USINF*"
  – Variante (CALL30): „UINF1*"

- CALL-DML kennt zwei Formate zur Übergabe von Namen, das Einzelnamenformat zur Angabe eines Namens und das Namensleistenformat zur Angabe mehrer Namen.
  - Einzelnamenformat

    Beispiel für Variante (CALL8):

    ```
    OF-SET ---> OF-SET␣␣
    ```

  - Namensleistenformat

    Die Namen werden durch Kommata getrennt und in runde Klammern eingeschlossen.

    Beispiel für Variante (CALL8):

    ```
    (INCREDIB)
    (BEISPIEL,BEISPIE␣,B-SPIEL␣,BE␣␣␣␣␣␣)
    ```

8.2.2 Format-Tabelle

Parameter	Inhalt	Länge bei Variante		Format
		(CALL8)	(CALL30)	
1. FCOD	-	6		Schlüsselwort
2. FOPT	-	6		Schlüsselwort
3. SOPT	RET, VAR	6/≥2		formatgebunden/formatfrei, siehe Seite 203
	RET, RES, LMS, TAL, SOA/SOD, NOW	12/≥2		
4. UINF	DBH-Kommunikation	126		siehe Seite 205
5. RECN	Satzname	8	30	Einzelnamenformat
	Satzname oder ␣␣␣␣␣␣␣␣	8	30	Einzelnamenformat oder Leerzeichen
	RECORD	8	30	RECORD␣␣/RECORD␣␣␣...
6. SETN	Setname	8	30	Einzelnamenformat
	1-25 Setnamen	10-226	32-776	Namensleistenformat
7. RLMN	Realm-Name	8	30	Einzelnamenformat
	Feldname bei der Zusatzwahl SOA oder SOD	8	30	Einzelnamenformat
	1-25 Realm-Namen	10-226	32-776	Namensleistenformat
	1-25 Feldnamen bei der Zusatzwahl SOA oder SOD	10-226	32-776	Namensleistenformat
8. ITMN	Feldname	8	30	Einzelnamenformat
	1-25 Feldnamen	10-226	32-776	Namensleistenformat
	Suchausdruck	-		siehe Seite 245
9. RECA[1]	vollständiger Satz	-		Subschemaformat
	Auswahl von Feldern ohne Angabe von VAR	-		Feldpositionen im Satzbereich wie im Subschemaformat
	Auswahl von Feldern mit Angabe von VAR	-		Satzformat entspricht der Feldnamenleiste, Länge = Summe der Feldlängen
10. SPP1	RETAINING ohne Setnamen	9		siehe Seite 209
	RETAINING mit Setnamen	17-235	39-785	
	Subschema (READYC)	30		Subschemaname in voller Länge mit Leerzeichen ergänzt

Tabelle 23: Formate der CALL-DML (Teil 1 von 2)

Parameter	Inhalt	Länge bei Variante		Format
		(CALL8)	(CALL30)	
11. SPP2	Ganzzahl (FIND4/FTCH4)	4		binäre Ganzzahl ≠ 0
	Ergebnissetname (FIND7A/FTCH7A)	8	30	Einzelnamenformat
	implizit def. Daten: DB-Key der LOCATION MODE-Klausel und/oder Realm-Name der WITHIN-Klausel (STORE1/STOR1L, STORE2/STOR2L, FIND2/FTCH2)	42		siehe Seite 210
12. SPP3	Begrenzungssetname (FIND7A/FTCH7A)	8	30	Einzelnamenformat

Tabelle 23: Formate der CALL-DML (Teil 2 von 2)

1 Wenn der Satzbereich zur Rückgabe von Daten verwendet wird, wird er in der Regel auch dann überschrieben, wenn die DML-Funktion FTCH oder GET mit Fehler abgebrochen wurde (Status ≠ 000). Wenn der Statuscode jedoch mit C, P oder S beginnt, bleibt der Satzbereich unverändert.

8.2.3 Format der Zusatzwahl (SOPT)

Mit diesem Parameter wählen Sie folgende Zusatzfunktionen:

RET	Retaining	
VAR	verkürztes Satzformat	
RES	Result-Set	FIND7A/FTCH7A
LMS	Limited-by-Dynamic-Set	
TAL	Tallying	
SOA/SOD	Sorted ascending/descending	
NOW	No Wait	READYC, FIND/FTCH

Dieser Parameter ist immer wahlfrei. Stehen mehr als eine der Zusatzfunktionen zur Auswahl, so können Sie jede mögliche Kombination dieser Funktionen wählen. Soll keine der zur Wahl stehenden Funktionen angewandt werden, so müssen Sie das Parameterfeld mit Leerzeichen (6-12) oder mit leeren Klammern versorgen.

Die Funktionen Retaining, Result-Set und Limited-by-Dynamic-Set sind mit der Verwendung eines zusätzlichen Parameters gekoppelt, nämlich Spezialparameter-1 (RET), -2 (RES) und -3 (LMS).
Diese Parameter müssen Sie nur dann angeben, wenn Sie die zugehörige Zusatzwahl angegeben haben. Sonst brauchen Sie diese Parameter in der Parameterleiste des CALL-DML-Aufrufs nicht anzuführen, sofern sie am Ende stehen.

Zur Übergabe der Zusatzwahl-Symbole im Parameterfeld „Zusatzwahl“ stehen zwei Darstellungsformen zur Verfügung.
Beide Darstellungsformen sind gleichermaßen dazu geeignet, das Parameterfeld „Zusatzwahl“ zur Programmlaufzeit zu erstellen bzw. zu modifizieren.

Künftige Erweiterungen der Zusatzwahl werden nur in die formatfreie Darstellung aufgenommen, um sicherzustellen, dass die Parameterfelder „Zusatzwahl“ in Anwenderprogrammen aufwärts kompatibel sind.

Formatgebundene Zusatzwahl

Der Parameter „Zusatzwahl“ hat eine feste Länge (6 bzw. 12 Zeichen). Jedes Symbol hat eine feste Position innerhalb des Parameters. Wünschen Sie eine Funktion nicht, müssen Sie an die entsprechenden Stelle Leerzeichen schreiben.

Syntax:

```
– alle außer FIND7A/FTCH7A   {RET}{VAR}
                             {␣␣␣}{␣␣␣}

– FIND7A/FTCH7A              {RET}{RES}{LMS}{TAL}
                             {␣␣␣}{␣␣␣}{␣␣␣}{␣␣␣}
```

Beispiele

1. ␣␣␣␣␣␣ oder RET␣␣␣ oder ␣␣␣VAR oder RETVAR
2. 12 Leerzeichen oder RETRESLMSTAL oder ␣␣␣RESLMS␣␣␣ usw.

Formatfreie Zusatzwahl

Der Parameter „Zusatzwahl“ hat keine feste Länge. Er beginnt im ersten Byte des Parameterfeldes mit einer öffnenden runden Klammer und endet mit einer schließenden runden Klammer. Die Funktionssymbole können in beliebiger Folge, aber ohne Wiederholungen, an beliebiger Stelle stehen. Sie können unmittelbar aufeinander folgen oder durch beliebig viele Leerzeichen und Kommata getrennt werden.

Syntax:

```
param  ::=  ({[{sym}][{␣}]}...)
            {␣␣␣}  {,}

sym    ::=  RET/VAR/RES/LMS/TAL/{SOA}/NOW
                                {SOD}
```

Beispiele

1. () oder (␣␣␣,␣␣␣)
2. (RET) oder (RET,␣␣␣)
3. (RETRESLMS␣␣␣) oder (␣␣␣,RES,RET,LMS)

   Die Beispiele einer Zeile haben jeweils gleiche Bedeutung; sie wurden willkürlich gewählt, um die Möglichkeiten des Formates zu verdeutlichen.

8.2.4 Format der Benutzerinformation (UINF)

Den Parameter Benutzerinformation verwenden alle CALL-DML-Aufrufe. Er identifiziert eine Transaktion gegenüber dem DBH. Bei einer Multi-DB-Anwendung wählt er außerdem die Datenbank aus, auf die ein DML-Aufruf zugreifen soll.

Welche Daten der Benutzerinformation Sie sichern müssen, ergibt sich im Einzelnen aus dem folgenden Abschnitt.

Während der Ausführung eines CALL-DML-Aufrufs dürfen Sie im gesamten Parameterbereich nichts ändern. Für dieselbe Transaktion kann selbstverständlich nur ein DML-Aufruf zur selben Zeit bearbeitet werden. Diese Einschränkung gilt auch für Programmroutinen, die außerhalb des normalen Ablaufs einer Transaktion liegen (Fehlerroutine, STXIT).

Das Parameterfeld der Benutzerinformation auf Wortgrenze auszurichten empfiehlt sich, wird aber nicht vorausgesetzt. Seine Länge beträgt 126 Bytes.

Der Benutzerinformationsbereich

Inhalt	Eingabe	Ausgabe	Länge	Distanz	Typ
System Communication Locations					
- Realm-Name		X	30	0	Character
- Satzname		X	30	30	Character
- Setname		X	30	60	Character
Ergebnisfelder					
- Anweisungscode		X	2	90	Character
- Statuscode		X	3	92	Character
Leerfeld			1	95	binär
DATABASE-KEY	X	X	4	96	binär
Zähler		X	4	100	binär
Leerfeld			7	104	binär
DB-Kennzeichen	X	X	1	111	binär
DATABASE-KEY-LONG	X	X	8	112	binär
Benutzerinformationsende	X		6	120	Character: USINF*/ UINF1*

Tabelle 24: Benutzerinformationsbereich

Regeln

Eingabe

DATABASE-KEY müssen Sie bei folgenden Anweisungen mit einem Database-Key-Wert vom Typ DATABASE-KEY vorbesetzen:

- FIND1/FTCH1
- ACCPTC mit Zusatzwahl RLMDBK

In allen anderen Fällen ist der Inhalt des Feldes als Eingabe belanglos.

Den Database-Key-Wert tragen Sie wie folgt in das Feld ein:

- Byte 1: Satzartnummer (REC-REF)
- Byte 2-4: Satzfolgenummer (RSQ)

Database-Key-Werte mit einer REC-REF > 254 und/oder einer RSQ > 2^{24}-1 können in diesem Feld nicht übergeben werden.

Leerfelder müssen Sie vor jedem READYC-Aufruf mit binär Null oder Leerzeichen löschen.

DB-Kennzeichen müssen Sie vor jedem READYC-Aufruf mit binär Null löschen. Nach dem READYC-Aufruf ist das Feld mit dem korrekten DB-Kennzeichen versorgt. Sie müssen es bei jedem CALL-DML-Aufruf wieder mitliefern (Ausnahme: FINISC), um seine zugehörige Verarbeitungskette und damit die Datenbank zu identifizieren.

DATABASE-KEY-LONG

müssen Sie bei folgenden Anweisungen mit einem Database-Key-Wert vom Typ DATABASE-KEY-LONG vorbesetzen:

- FIND1L/FTCH1L
- ACCPTL mit Zusatzwahl RLMDBK

In allen anderen Fällen ist der Inhalt des Feldes als Eingabe belanglos.

Den Database-Key-Wert tragen Sie wie folgt in das Feld ein:

- Byte 1-2: Satzartnummer (REC-REF)
- Byte 3-4: binär 0
- Byte 5-8: Satzfolgenummer (RSQ)

Benutzerinformationsende

müssen Sie mit der Zeichenfolge USINF* besetzen, wenn Sie die Variante (CALL8) verwenden, bzw. mit der Zeichenfolge UINF1*, wenn Sie die Variante (CALL30) verwenden. Diese Kennzeichen identifizieren das Parameterfeld „Benutzerinformation“, um sicherzustellen, dass über den Statuscode Rückmeldungen an das Anwender-Programm möglich sind.
Erkennt der CALL-DML-Umsetzer das Schlüsselwort USINF* bzw. UINF1* nicht, versucht er, den Fehlerausgang DSCEXT anzuspringen, der in jedem CALL-DML-Anwenderprogramm als Einsprungadresse (ENTRY) definiert sein muss (siehe Seite 124). Der DML-Aufruf wird in dem Fall nicht ausgeführt.

In openUTM-Anwenderprogrammen wird die DSCEXT-Routine nicht bedient.

Ausgabe

System Communication Locations

Die Funktion der Felder entspricht denen der COBOL-DML.

Realmname

Ergebnisfeld beim ACCPTC bzw. ACCPTL; bei den Funktionswahlen RLM... legt der CALL DML-Umsetzer hier den gesuchten Realm-Namen ab.

Statuscode

Zusätzlich zu den Statuscodes, die auch bei COBOL-DML vorkommen, kennt die CALL-DML noch weitere Statuscodes (siehe Seite 380).

Bei einem Statuscode der CALL-DML enthalten die Felder der „System Communication Locations“ keine signifikanten Angaben.

DATABASE-KEY

Rückgabe des Database-Key-Wertes bei ACCPTC mit Funktionsauswahl DB-KEY und DBK...

Der Database-Key-Wert wird wie folgt zurückgegeben:

– Byte 1: Satzartnummer (REC-REF)
– Byte 2-4: Satzfolgenummer (RSQ)

Database-Key-Werte mit einer REC-REF > 254 und/oder einer RSQ > 2^{24}-1 kann UDS/SQL nicht in diesem Feld abliefern und meldet in diesem Fall den DATABASE-STATUS 15102.

Zähler

enthält nach Ausführung eines FIND7A/FTCH7A mit Funktionswahl ...FST oder ...SEX (Suchausdruck) und Zusatzwahl TAL (Tallying-Funktion) die Anzahl aller Sätze der Auswahlmenge, die die Suchbedingungen erfüllen.

DATABASE-KEY-LONG

Rückgabe eines Database-Key-Wertes bei ACCPTL mit Funktionswahl DB-KEY und DBK...

Der Database-Key-Wert wird wie folgt zurückgegeben:

– Byte 1-2: Satzartnummer (REC-REF)
– Byte 3-4: binär 0
– Byte 5-8: Satzfolgenummer (RSQ)

8.2.5 Format des Spezialparameter-1 (SPP1)

Wollen Sie RETAINING anwenden, so müssen Sie im Spezialparameter-1 die gewünschten Funktionen spezifizieren. Folgende Angaben sind - analog zur COBOL-DML - möglich:

```
MULTIPLE␣      RETAINING CURRENCY FOR MULTIPLE
RLM            RETAINING CURRENCY FOR REALM
REC            RETAINING CURRENCY FOR RECORD
SET            RETAINING CURRENCY FOR SETS
STNsetname     RETAINING CURRENCY FOR setname-1,...
STEsetname     RETAINING CURRENCY FOR ALL SETS EXCEPT setname-1,...
```

Format

Die Übergabe dieser Symbole erfolgt im festen Format (mindestens 9 Bytes):

Bytes 1...9	**Bytes 10...235 bei Variante (CALL8)** **Bytes 10...785 bei Variante (CALL30)**
M U L T I P L E ␣ R L M R E C S E T R L M R E C ␣ ␣ ␣ ␣ ␣ ␣ R E C S E T R L M ␣ ␣ ␣ S E T R L M ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ R E C ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ S E T	
R L M R E C S T N R L M ␣ ␣ ␣ S T N ␣ ␣ ␣ R E C S T N ␣ ␣ ␣ ␣ ␣ ␣ S T N	{ setname setname-1,... }
R L M R E C S T E R L M ␣ ␣ ␣ S T E ␣ ␣ ␣ R E C S T E ␣ ␣ ␣ ␣ ␣ ␣ S T E	{ setname setname-1,... }

Tabelle 25: Übergabeformat des Spezialparameter-1

Setnamen können Sie nur bei Schlüssel STN oder STE angeben! Sie können max. 25 Setnamen angeben. Wenn in den Bytes 7 bis 9 nicht STN oder STE steht, werden die Bytes 10 bis 235 bzw. 10 bis 785 nicht ausgewertet. Die Funktion STE gibt es nur in der CALL-DML.

8.2.6 Format des Spezialparameter-2 (SPP2)

Wollen Sie die Funktionen STORE1/STOR1L bzw. STORE2/STOR2L mit der Funktionswahl IMPDAT verwenden, so müssen Sie im Spezialparameter-2 angeben:

– Database-Key-Wert (wenn LOCATION MODE DIRECT bzw. LOCATION MODE DIRECT LONG für die betreffende Satzart vereinbart ist) und/oder

– Realm-Name (wenn in der WITHIN-Klausel der betreffenden Satzart mehrere Realm-Namen angegeben sind und die Satzart nicht Membersatzart einer verteilbaren Liste ist)

Wollen Sie die Funktionen FIND2/FTCH2 mit der Funktionswahl ANYIMP verwenden, so müssen Sie im Spezialparameter-2 den Namen des Realm angeben, in dem der Satz gesucht werden soll.

Format

Byte 1...4	**Byte 5...34**	**Byte 35...42**
Database-Key-Wert vom Typ DATABASE-KEY (relevant für STORE1, STORE2); falls nicht benutzt: binär 0	Realm-Name in voller Länge, ergänzt durch Leerzeichen (relevant für FIND2/FTCH2 sowie STORE1/STOR1L und STORE2/STOR2L); falls nicht benutzt: Leerzeichen	Database-Key-Wert vom Typ DATABASE-KEY-LONG (relevant für STOR1L, STOR2L); falls nicht benutzt: binär 0

Tabelle 26: Übergabeformat des Spezialparameter-2

8.3 Die CALL-DML-Aufrufe

8.3.1 Übersicht über die CALL-DML-Funktionen

Die grauen Hinterlegungen in der Übersicht bedeuten:
Wenn Sie die CALL-DML-Funktion ausgeführt haben, sind hier Informationen abgelegt.

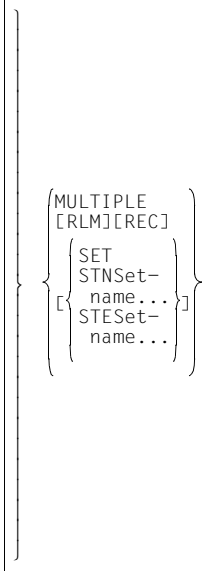
1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN	9. RECA
ACCPTC	DB-KEY DBKREC DBKRLM DBKSET RLMNAM RLMREC RLMSET RLMDBK		DB-Key DB-Key DB-Key DB-Key Realmname Realmname Realmname DB-Key, Realmname	– Satzname – – – Satzname – –	– – – Setname – – Setname –	– – Realmname – – – – –		
ACCPTL	DB-KEY DBKREC DBKRLM DBKSET RLMNAM RLMREC RLMSET RLMDBK		DB-Key DB-Key DB-Key DB-Key Realmname Realmname Realmname DB-Key, Realmname	– Satzname – – – Satzname – –	– – – Setname – – Setname –	– – Realmname – – – – –		
CONNEC	TO-ALL TO-SET	} [RET]		[Satzname] [Satzname]	– Setname...			
DISCON	FRMALL FRMSET ALLFRM			[Satzname] [Satzname] –	– Setname... Setname...			
ERASEC	CORUNT PERMAN SELTIV ALLMEM			} Satzname				
FINISC	ALLRLM ALLCAN							
FREEC	ALLREC CORUNT							

10. SPP1	11. SPP2	12. SPP3	Funktion
			überträgt – den DATABASE-KEY-Wert des CRR, CRS, CRA bzw. CRU in den Benutzerinformationsbereich (siehe Seite 205). – den Namen des Realm, in dem der CRR, CRS, CRU bzw. der zu einem angegebenen Database-Key-Wert gehörige Satz gespeichert ist.
			überträgt – den DATABASE-KEY-LONG-Wert des CRR, CRS, CRA bzw. CRU in den Benutzerinformationsbereich (siehe Seite 205). – den Namen des Realm, in dem der CRR, CRS, CRU bzw. der zu einem angegebenen Database-Key-Wert gehörige Satz gespeichert ist.
bei RET: {SET {STNSetname..} {STESetname..}			hängt den CRU in Set-Occurrences ein
			– löst den CRU aus Set-Occurrences – entfernt alle Sätze aus dynamischen Sets
			löscht den CRU ggf. mit zugehörigen Membersätzen aus der Datenbank
			beendet eine Transaktion und gibt gesperrte Realms und Seiten frei
			beendet die Wirkung des KEEP-Status

1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN	9. RECA
FIND1	[DBKPRI] [DBKNXT]	[RET] [NOW]	DB-Key	[Satzname]				
FIND1L	[DBKPRI] [DBKNXT]	[RET] [NOW]	DB-Key	[Satzname]				
FIND2	ANYREC ANYIMP DUPLIC	[RET] [NOW]		Satzname				Feldinhalt Feldinhalt -
FIND3	SETNAM SETITM RECNAM RECITM	[RET] [NOW]		- Satzname	Setname Setname - -		- Feldname... - Feldname...	
FIND4	SETNXT SETPRI SETFST SETLST SETSPC	[RET] [NOW]		[{Satzname / RECORD␣␣}]	Setname			
	RLMNXT RLMPRI RLMFST RLMLST RLMSPC	[RET] [NOW]		[{Satzname / RECORD␣␣}]		Realm- name		
	RECNXT RECPRI RECFST RECLST RECSPC	[RET] [NOW]		Satzname				
FIND5	CORUNT RECNAM RECSET SETNAM RECRLM RLMNAM	[RET] [NOW]		- Satzname Satzname - Satzname -	- - Setname Setname - -	- - - - Realmname Realmname		
FIND6		[RET] [NOW]			Setname			

10. SPP1	11. SPP2	12. SPP3	Funktion
Bei RET:			Zugriff über einen Database-Key-Wert vom Typ DATABASE-KEY
			Zugriff über einen Database-Key-Wert vom Typ DATABASE-KEY-LONG
	– impl. def. Datenbereich –		Zugriff über einen Calc-Key (Hashverfahren)
			Zugriff auf einen Satz, der in bestimmten Feldinhalten mit dem CRR bzw. CRS übereinstimmt oder Zugriff auf einen Satz, der einem vorhergehend abgearbeiteten Suchausdruck (FIND7A/FTCH7A) genügt.
{ MULTIPLE [RLM][REC] [{ SET / STNSet-name... / STESet-name... }] }	– – – – pos. Ganzzahl		Zugriff auf den letzten oder ersten Satz, auf den Nachfolger oder Vorgänger des CRR, CRS bzw. CRA oder auf einen Satz, dessen Position einem anzugebenden Zahlenwert entspricht innerhalb einer Auswahlmenge. Die Auswahlmenge kann eine Satzart, eine Set-Occurence, ein Realm oder die Durchschnittsmenge einer Satzart mit einem Realm sein.
	– – – – pos. Ganzzahl		
	– – – – pos. Ganzzahl		
			Zugriff auf den CRR, CRS, CRA bzw. CRU
			Zugriff auf den Ownersatz eines CRS

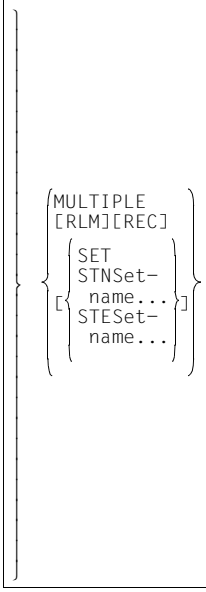
1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN	9. RECA
FIND7A	RECFST	[RET] [NOW] [RES]	bei TAL:		–	Feldname...		
	SELFST	[LMS] [TAL]		Satzname	Setname	Feldname...		
	CURFST	[SOA] [SOD]			Setname	Feldname...		
	RECSEX	[RET] [NOW] [RES] [LMS] [TAL]	Satz- anzahl		–	Feldname...		
					Setname	Feldname...		
	CURSEX	[SOA] [SOD]		Satzname	Setname	Feldname...	Such- ausdruck	
	RECITM RECITN SELITM SELITN SELITP CURITM CURITN CURITP	[RET] [NOW]		Satzname	– Setname Setname		Feld- name...	Feld- inhalt

10. SPP1	11. SPP2	12. SPP3	Funktion
	bei RES: Ergebnis- setname	bei LMS: Begren- zungs- setname	Zugriff auf Sätze über beliebige Felder, ggf. Zählen und Zwischenspeichern der Treffersätze und Suchen mit Maske

1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN	9. RECA
FTCH1	[DBKPRI] [DBKNXT]	[RET] [NOW]	DB-Key	[Satzname]				Satzinhalt
FTCH1L	[DBKPRI] [DBKNXT]	[RET] [NOW]	DB-Key	[Satzname]				Satzinhalt
FTCH2	ANYREC ANYIMP DUPLIC	[RET] [NOW]		Satzname				Feldinhalt Satzinhalt Feldinhalt Satzinhalt Satzinhalt
FTCH3	SETNAM SETITM RECNAM RECITM	[RET] [NOW]		Satzname	Setname Setname - -		- Feldname... Feldname...	Satz- inhalt
FTCH4	SETNXT SETPRI SETFST SETLST SETSPC	[RET] [NOW]		[Satzname RECORD␣␣]	Setname			
	RLMNXT RLMPRI RLMFST RLMLST RLMSPC	[RET] [NOW]		[Satzname RECORD␣␣]		Realm- name		
	RECNXT RECPRI RECFST RECLST RECSPC	[RET] [NOW]		Satzname				
FTCH5	CORUNT RECNAM RECSET SETNAM RECRLM RLMNAM	[RET] [NOW]		- Satzname Satzname - Satzname -	- - Setname Setname - -	- - - - Realmname Realmname		
FTCH6		[RET] [NOW]			Setname			

10. SPP1	11. SPP2	12. SPP3	Funktion
			Zugriff über einen Database-Key-Wert vom Typ DATABASE-KEY
			Zugriff über einen Database-Key-Wert vom Typ DATABASE-KEY-LONG
	– impl. def. Datenbereich –		Zugriff über einen Calc-Key (Hashverfahren)
MULTIPLE [RLM][REC] [SET / STNSet-name... / STESet-name...]			Zugriff auf einen Satz, der in bestimmten Feldinhalten mit dem CRR bzw. CRS übereinstimmt oder Zugriff auf einen Satz, der einem vohergehend abgearbeiteten Suchausdruck (FIND7A/FTCH7A) genügt
	– – – – pos. Ganzzahl		Zugriff auf den letzten oder ersten Satz, auf den Nachfolger oder Vorgänger des CRR, CRS bzw. CRA oder auf einen Satz, dessen Position einem anzugebenden Zahlenwert entspricht innerhalb einer Auswahlmenge. Die Auswahlmenge kann eine Satzart, eine Set-Occurrence, ein Realm oder die Durchschnittsmenge einer Satzart mit einem Realm sein.
	– – – – pos. Ganzzahl		
	– – – – pos. Ganzzahl		
			Zugriff auf den CRR, CRS, CRA bzw. CRU
			Zugriff auf den Ownersatz eines CRS

1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN	9. RECA
FTCH7A	RECFST	[RET] [NOW] [RES]	bei TAL:		–	Feldname...		Satz- inhalt
	SELFST	[LMS] [TAL]		Satzname	Setname	Feldname...		
	CURFST	[SOA] [SOD]	Satz- anzahl		Setname	Feldname...		
	RECSEX	[RET] [NOW] [RES]			–	Feldname...		
		[LMS] [TAL]		Satzname	Setname	Feldname...	Such- ausdruck	
	CURSEX	[SOA] [SOD]			Setname	Feldname...		
	RECITM RECITN SELITM SELITN SELITP CURITM CURITN CURITP	[RET] [NOW]		Satzname	– Setname Setname		Feld- name-1...	Feld- inhalt/ Satz- inhalt

10. SPP1	11. SPP2	12. SPP3	Funktion
	bei RES: Ergebnis- setname	bei LMS: Begren- zungs- setname	Zugriff auf Sätze über beliebige Felder, ggf. Zählen und Zwischenspeichern der Treffersätze und Suchen mit Maske

1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN	9. RECA
GETC	CORUNT ITMNAM	– [VAR]		[Satzname] Satzname			– Feld– name–1...	Satzinhalt Feldinhalt
IFC	OWNALL OWNSET MEMALL MEMSET TENALL TENSET EMPTYS				– Setname – Setname – Setname Setname			
KEEPC								
MODIF1	CORUNT INCALL ONLALL INCSET ONLSET	} [RET]		} Satzname	– – – Setname... Setname...			Satzinhalt Satzinhalt – Satzinhalt –
MODIF2	CORUNT INCALL INCSET	} [RET] } [VAR]		} Satzname	– – Setname		} Feld– } name...	} Feld– } inhalt
READYC	ALLRTR ALLUPD ALLPRT ALLPUP ALLERT ALLEUP RLMRTR RLMUPD RLMPRT RLMPUP RLMERT RLMEUP	} [NOW]				– – – – – – } Realm– } name...		

10. SPP1	11. SPP2	12. SPP3	Funktion
			stellt den CRU oder einzelne Felder des CRU im Satzbereich zur Verfügung
			prüft Set-Mitgliedschaften
			schützt den CRU vor demZugriff durch andere Transaktionen bis zu einer FREE-Anweisung oder dem Ende der Transaktion
bei RET: SET STNSetname.. STESetname..			ändert Satzinhalt oder Feldinhalte des CRU und/oder hängt ihn innerhalb eines Set in eine andere Set-Occurrence um
Subschema-name	wird nicht mehr benötigt		eröffnet eine Transaktion oder eine Verarbeitungskette

1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN	9. RECA
STORE1	RECNAM IMPDAT	[RET]		Satzname				Satz- inhalt
STOR1L	RECNAM IMPDAT	[RET]		Satzname				Satz- inhalt
STORE2	ITMNAM IMPDAT	[RET] [VAR]		Satzname			Feld- name...	Feld- inhalt
STOR2L	ITMNAM IMPDAT	[RET] [VAR]		Satzname			Feld- name...	Feld- inhalt

10. SPP1	11. SPP2	12. SPP3	Funktion
bei RET: { MULTIPLE [RLM][REC] { SET / STNSet-name... / STESet-name... } [ ] }	– impl. def. Datenbereich –		– überträgt einen Satz oder einzelne Felder oder komprimierte Sätze aus der UWA als neuen Satz in die Datenbank – fügt den neuen Satz in alle Sets ein, für die seine Satzart im Schema als AUTOMATIC Member definiert ist. – richtet eine neue Set-Occurrence für jeden Set ein, für den die Satzart im Schema als Ownersatzart definiert ist Wenn Sie bei Funktionswahl IMPDAT einen Database-Key-Wert mit einer REC-REF > 254 und/oder einer RSQ > 2^{24}-1 angeben wollen, müssen Sie STOR1L bzw. STOR2L verwenden.
	– impl. def. Datenbereich –		

8.3.2 Die Funktionen der CALL-DML

Die Funktion eines Aufrufs wird durch die Parameter Funktionsname und Funktionswahl festgelegt.

Die Regeln der Ausführung eines CALL-DML-Aufrufs entsprechen denen der COBOL-DML (siehe Abschnitt „Anweisungen der COBOL-DML“ auf Seite 139). Sie werden deshalb hier nicht noch einmal aufgeführt.
Um die entsprechenden Regeln der COBOL-DML leicht auffinden zu können, sind den Schlüsselwörtern des Parameters „Funktionswahl“ die entsprechenden COBOL-DML-Anweisungen gegenübergestellt.
Abweichungen zur COBOL-DML werden in jedem Fall genau beschrieben.

In der Regel hängt es von der getroffenen Funktionswahl ab, ob und in welcher Weise ein Parameter verwendet wird. Deshalb werden bei der Beschreibung der Parameter die Funktionswahl-Schlüsselwörter angegeben, für die diese Beschreibung zutrifft.

Beispiel

```
Funktionsname:          ACCPTC
                        .
                        .
                        .
Setname:                DBKSET/RLMSET          Eingabe eines Setnamens
```

Der Parameter Setname wird nur bei den beiden angegebenen Funktionswahlen verwendet, bei den übrigen nicht.

Wird jedoch kein Funktionswahl-Schlüsselwort angegeben, so trifft die Beschreibung bei allen Möglichkeiten zu.

Im Folgenden sind die Formate der CALL-DML-Aufrufe ausführlich beschrieben.

Informationen der Currency-Tabelle sicherstellen (ACCPTC, ACCPTL)

Die Funktionen ACCPTC und ACCPTL ermitteln

– den Inhalt der angegebenen Currency-Information (Database-Key-Wert),
– den zu einem bestimmten Database-Key-Wert gehörenden Realm-Namen.

ACCPTC und ACCPTL unterscheiden sich hinsichtlich der Rückgabe bzw. Eingabe des Database-Key-Wertes.

ACCPTL kann nur in Verbindung mit einem SSITAB-Modul einer Version ab UDS/SQL V2.0 ausgeführt werden (siehe Handbuch „Aufbauen und Umstrukturieren“). Andernfalls meldet UDS/SQL den Statuscode C98.

Funktionsname: **ACCPTC**

Funktionsname: **ACCPTL**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
DB-KEY	ACCEPT *feldname* FROM CURRENCY
DBKREC	ACCEPT *feldname* FROM *satzname* CURRENCY
DBKRLM	ACCEPT *feldname* FROM *realmname* CURRENCY
DBKSET	ACCEPT *feldname* FROM *setname* CURRENCY
RLMNAM	ACCEPT *feldname* FROM REALM-NAME
RLMREC	ACCEPT *feldname* FROM *satzname* REALM-NAME
RLMSET	ACCEPT *feldname* FROM *setname* REALM-NAME
RLMDBK	ACCEPT *feldname* FROM *feldname* REALM-NAME

Benutzerinformation:	DB-KEY/DBK... Rückgabe des Database-Key-Wertes. Bei ACCPTC kann UDS/SQL nur Database-Key-Werte mit einer REC-REF ≤ 254 und einer RSQ ≤ 2^{24}-1 zurückgeben. Näheres siehe Seite 205-207.
	RLM... Rückgabe des Realm-Namens im Feld Realm-Name der System Communication Locations
	RLMDBK Eingabe eines Database-Key-Wertes. Bei ACCPTC können Sie nur Database-Key-Werte mit einer REC-REF ≤ 254 und einer RSQ ≤ 2^{24}-1 angeben. Näheres siehe Seite 205-207.
	Rückgabe des DATABASE-STATUS
Satzname:	DBKREC/RLMREC: Eingabe eines Satznamens
Setname:	DBKSET/RLMSET Eingabe eines Setnamens
Realmname:	DBKRLM Eingabe eines Realm-Namens

Herstellen von Setverbindungen (CONNEC)

Die CONNEC-Funktion sorgt dafür, dass der Current Record of Rununit (CRU) Member wird in einem oder mehreren Sets, in denen seine Satzart wahlfreier Member ist (d.h. MANDATORY MANUAL, OPTIONAL AUTOMATIC oder OPTIONAL MANUAL).

Wahlweise können Sie durch die Verwendung der RETAINING-Angabe verhindern, dass der Satz für alle oder für die angegebenen Sets, in die er eingefügt wird, Current Record of Set (CRS) wird.

Funktionsname: **CONNEC**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
TO-ALL	CONNECT[*satzname*] TO ALL
TO-SET	CONNECT[*satzname*] TO *setname-1*,...

Zusatzwahl: Eingabe von RET (wahlfrei)

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: Eingabe des Satznamens (wahlfrei)

Setname: TO-SET
Eingabe eines oder mehrerer Setnamen

Spezialparameter-1: Eingabe von Retaining-Parametern bei Zusatzwahl RET; bei CONNEC ist Retaining nur für Sets möglich.

Lösen von bestehenden Setverbindungen (DISCON)

Die DISCON-Funktion beendet die Mitgliedschaft des Current Record of Rununit (CRU) in einem oder mehreren Sets, in denen er zur Zeit Member ist, vorausgesetzt, dass seine Satzart zum OPTIONAL Member erklärt ist. Die Mitgliedschaft aller Sätze der Current Set-Occurrence (nur bei dynamischen Sets) können Sie mit ALLFRM beenden.

Funktionsname: **DISCON**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
FRMALL	DISCONNECT[*satzname*] FROM ALL
FRMSET	DISCONNECT[*satzname*] FROM *setname-1*,...
ALLFRM	DISCONNECT ALL FROM *setname-2*,...

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: FRMALL/FRMSET
Eingabe eines Satznamens (wahlfrei)

Setname: FRMSET/ALLFRM
Eingabe eines oder mehrerer Setnamen

Löschen von Sätzen und deren Setverbindungen (ERASEC)

Die ERASEC-Funktion löscht den CRU aus der Datenbank und entfernt ihn aus allen Set-Occurrences, in denen er Member war. Zusätzlich kann die ERASEC-Funktion

– alle MANDATORY-Membersätze des CRU löschen oder

– alle Sätze, die OPTIONAL Member sind in Set-Occurrences, in denen der betreffende Satz der Owner ist, aus diesen Set-Occurrences entfernen und sie wahlweise löschen.

Funktionsname: **ERASEC**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
CORUNT	ERASE *satzname*
PERMAN	ERASE *satzname* PERMANENT MEMBERS
SELTIV	ERASE *satzname* SELECTIVE MEMBERS
ALLMEM	ERASE *satzname* ALL MEMBERS

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: Eingabe des Satznamens

Wiedergewinnen von Daten (FIND/FTCH)

Die FIND/FTCH-Funktionen machen einen durch einen Satzauswahlausdruck gefundenen Satz zum

– Current Record of Rununit (CRU),

– Current Record of Area (CRA) des Realm, in dem er gespeichert ist,

– Current Record of Record (CRR) seiner Satzart und

– Current Record of Set (CRS) für alle Sets, in denen er der Owner oder ein Member ist.

Die FTCH-Funktionen übertragen außerdem den Inhalt des gefundenen Satzes, wie er im Subschema definiert ist, in den Satzbereich.

Wahlweise können Sie durch Angabe von RET im Parameter „Zusatzwahl“ verlangen, dass von der FIND/FTCH-Funktion betroffene Informationen der Currency-Tabelle nicht verändert werden.

Wahlweise können Sie durch Angabe von NOW im Parameter „Zusatzwahl“ verlangen, dass beim Zugriff auf eine Seite, die von einer Transaktion gesperrt ist, nicht gewartet, sondern die Kontrolle an das Anwenderprogramm zurückgegeben wird (DATABASE-STATUS 04020).

Die FIND/FTCH-Funktionen können Sie mit jeweils 7 verschiedenen Funktionsnamen ansprechen. Jeder Funktionsname entspricht einem der 7 Formate der COBOL-DML. Die Versorgung der Parameter 2 bis 12 ist bei FIND und FTCH identisch.

Die Funktionen FIND1/FTCH1 bzw. FIND1L/FTCH1L greifen über einen Database-Key-Wert auf einen Satz zu. FIND1/FTCH1 und FIND1L/FTCH1L unterscheiden sich hinsichtlich des einzugebenden Database-Key-Wertes.

FIND1L und FTCH1L können nur in Verbindung mit einem SSITAB-Modul einer Version ab UDS/SQL V2.0 ausgeführt werden (siehe Handbuch „Aufbauen und Umstrukturieren“). Andernfalls meldet UDS/SQL den Statuscode C98.

Funktionsname: **FIND1/FTCH1**

Funktionsname: **FIND1L/FTCH1L**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
	FIND *satzname* DATABASE-KEY IS *feldname*
DBKPRI	FIND *satzname* DATABASE-KEY IS *feldname* OR PRIOR
DBKNXT	FIND *satzname* DATABASE-KEY IS *feldname* OR NEXT

Zusatzwahl: Eingabe von RET (wahlfrei), NOW (wahlfrei)

Benutzerinformation: Eingabe des Database-Key-Wertes des Satzes, der gesucht werden soll.
Bei Find1/FTCH1 können Sie nur Database-Key-Werte mit einer REC-REF ≤ 254 und einer RSQ ≤ 2^{24}-1 angeben. Näheres siehe Seite 205-207.

Rückgabe des DATABASE-STATUS

Satzname: Eingabe eines Satznamens (wahlfrei)

Satzbereich: DBKPRI/DBKNXT
Bei Angabe von DBKPRI bzw. DBKNXT wird der Satz mit dem nächstkleineren bzw. nächstgrößeren Database-Key geliefert, falls es keinen Satz mit dem angegebenen Database-Key gibt. Bei jeder anderen Angabe wird die OR PRIOR/NEXT-Funktionalität nicht angewendet.

FTCH1
Rückgabe des gesuchten Satzes

Spezialparameter-1: Eingabe von Retaining-Parametern bei Zusatzwahl RET

Die FIND2/FTCH2-Funktion greift direkt über den CALC-Key zu.

ANYIMP muss dann verwendet werden, wenn die Satzart auf mehrere Realms verteilt ist.

Funktionsname: **FIND2/FTCH2**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
ANYREC	FIND ANY *satzname*
ANYIMP	FIND ANY *satzname*
DUPLIC	FIND DUPLICATE *satzname*

Zusatzwahl: Eingabe von RET (wahlfrei), NOW (wahlfrei)

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: Eingabe eines Satznamens; die Satzart muss mit LOCATION MODE CALC definiert sein.

Satzbereich: ANYREC/ANYIMP
Eingabe des CALC-Key der LOCATION MODE-Klausel an der Stelle im Satzbereich, die dem Subschemaformat des Satzes entspricht

DUPLIC
eine Eingabe des CALC-Key nötig; der DBH sucht einen Satz im Realm des CRR, der denselben CALC-Key-Wert hat wie der CRR.

FTCH2
Rückgabe des Satzes

Spezialparameter-1: Eingabe von Retaining-Parametern bei Zusatzwahl RET

Spezialparameter-2: ANYIMP
Eingabe des Realm-Namens, in dem der Satz gesucht werden soll:

Byte 1-4: nicht benutzt

Byte 5-34: Realm-Name in voller Länge mit Leerzeichen ergänzt.

Der Inhalt von Byte 5-34 wird nur ausgewertet, wenn die Satzart *satzname* auf mehrere Realms verteilt ist (auf Grund einer entsprechenden WITHIN-Klausel der Schema-DDL-Definition für die Satzart *satzname*).

Die FIND3/FTCH3-Funktion greift auf einen Satz zu, der in bestimmten Feldinhalten mit dem CRR bzw. CRS übereinstimmt oder der einem vorher abgearbeiteten Suchausdruck genügt.

Funktionsname: **FIND3/FTCH3**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
SETNAM	FIND DUPLICATE WITHIN *setname*
SETITM	FIND DUPLICATE WITHIN *setname* USING *feldname-1*,...
RECNAM	FIND DUPLICATE WITHIN satzname
RECITM	FIND DUPLICATE WITHIN *satzname* USING *feldname-1*,....

Zusatzwahl: Eingabe von RET (wahlfrei), NOW (wahlfrei)

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: SETITM/RECNAM/RECITM
Eingabe eines Satznamens; bei SETITM ist dies zur Identifikation der Felder erforderlich.

Setname: SET...
Eingabe eines Setnamens

Feldname: ...ITM
Eingabe eines oder mehrerer Feldnamen

Satzbereich: FTCH3
Rückgabe des Satzes

Spezialparameter-1: Eingabe von Retaining-Parametern bei Zusatzwahl RET

Die FIND4/FTCH4-Funktion greift auf den ersten oder letzten Satz, auf den Nachfolger oder Vorgänger des CRR, CRS bzw. CRA, oder auf einen Satz zu, dessen Position einem anzugebenden Zahlenwert innerhalb einer Auswahlmenge entspricht.

Funktionsname: **FIND4/FTCH4**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
SET...	FIND ..... { *satzname* | RECORD } WITHIN *setname*
RLM...	FIND ..... { *satzname* | RECORD } WITHIN *realmname*
REC...	FIND ..... *satzname*
...NXT ...PRI ...FST ...LST	FIND NEXT ..... FIND PRIOR ..... FIND FIRST ..... FIND LAST .....
...SPC	FIND { *ganzzahl* | *feldname* }

Alle 15 möglichen Kombinationen sind zugelassen und entsprechen in vollem Umfang der COBOL-DML.

Zusatzwahl: Eingabe von RET (wahlfrei), NOW (wahlfrei)

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname:	Eingabe eines Satznamens; SET.../RLM... Die Eingabe eines Satznamens ist wahlfrei; Eingabe von Leerzeichen bewirkt dasselbe wie Eingabe von RECORD␣␣. SET... Bei Eingabe eines Satznamens prüft der DBH zusätzlich, ob die angegebene Satzart die Membersatzart in dem genannten Set ist. RLM... Die Eingabe eines Satznamens schränkt die Auswahlmenge auf diese Satzart ein; andernfalls berücksichtigt der DBH alle Satzarten des Subschemas, die in dem Realm gespeichert sind.
Setname:	SET... Eingabe eines Setnamens
Realmname:	RLM... Eingabe eines Realm-Namens
Satzbereich:	FTCH4 Rückgabe des Satzes
Spezialparameter-1:	Eingabe von Retaining-Parametern bei Zusatzwahl RET
Spezialparameter-2:	...SPC Eingabe einer Ganzzahl. Die Zahl muss als 4 byte lange Binärzahl codiert sein; negative Werte sind zulässig, 0 nicht.

Die FIND5/FTCH5-Funktion greift auf den CRR, CRS, CRA bzw. CRU zu.

Funktionsname: **FIND5/FTCH5**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
CORUNT	FIND CURRENT
RECNAM	FIND CURRENT *satzname*
RECSET	FIND CURRENT *satzname* WITHIN *setname*
SETNAM	FIND CURRENT WITHIN *setname*
RECRLM	FIND CURRENT *satzname* WITHIN *realmname*
RLMNAM	FIND CURRENT WITHIN *realmname*

Zusatzwahl: Eingabe von RET (wahlfrei), NOW (wahlfrei)

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: REC...
Eingabe eines Satznamens; die Auswahlmenge wird auf die angegebene Satzart eingeschränkt, d.h. der DBH prüft, ob der CRS bzw. CRA von der angegebenen Satzart ist.

Setname: RECSET/SETNAM
Eingabe eines Setnamens

Realmname: RECRLM/RLMNAM
Eingabe eines Realm-Namens

Satzbereich: FTCH5
Rückgabe des Satzes

Spezialparameter-1: Eingabe von Retaining-Parametern bei Zusatzwahl RET (nur für betroffene Sets)

Die FIND6/FTCH6-Funktion greift auf den Ownersatz eines CRS zu.

Funktionsname:	**FIND6/FTCH6**
Funktionswahl:	entfällt; das entsprechende COBOL-DML-Format lautet: `FIND OWNER WITHIN` *`setname`*
Zusatzwahl:	Eingabe von RET (wahlfrei), NOW (wahlfrei)
Benutzerinformation:	Rückgabe des DATABASE-STATUS
Setname:	Eingabe eines Setnamens
Satzbereich:	FTCH6 Rückgabe des Ownersatzes
Spezialparameter-1:	Eingabe von Retaining-Parametern bei Zusatzwahl RET (nur für betroffene Sets)

Die FIND7A/FTCH7A-Funktion greift auf Sätze über beliebige Felder zu; ggf. werden Treffersätze gezählt und zwischengespeichert und mit Maske gesucht.

Funktionsname: **FIND7A/FTCH7A**

Funktionswahl:

CALL-DML	**entsprechende COBOL-DML-Anweisung**
RECFST	FIND *satzname*
CURFST	FIND *satzname* WITHIN *setname* CURRENT
SELFST	FIND *satzname* WITHIN *setname*
RECSEX	FIND *satzname* USING *suchausdruck*
CURSEX	FIND *satzname* WITHIN *setname* CURRENT USING *suchausdruck*
SELSEX	FIND *satzname* WITHIN *setname* CURRENT USING *suchausdruck*
RECITM	FIND *satzname feldname-1*
CURITM	FIND *satzname* WITHIN *setname* CURRENT USING *feldname-1*,...
SELITM	FIND *satzname* WHITHIN *setname* USING *feldname-1*,...
RECITN	FIND *satzname* USING *feldname-1*,... OR NEXT
CURITN	FIND *satzname* WITHIN *setname* CURRENT USING *feldname-1*,... OR NEXT
SELITN	FIND *satzname* WITHIN *setname* USING *feldname-1*,... OR NEXT
CURITP	FIND *satzname* WITHIN *setname* CURRENT USING *feldname-1*,... OR PRIOR
SELITP	FIND *satzname* WITHIN *setname* USING *feldname-1*,... OR PRIOR

Zusatzwahl:	wahlweise Eingabe von RET für Retaining-Funktionen RES wenn Sie einen dynamischen Ergebnis-Set angeben wollen (Spezialparameter-2) LMS wenn Sie einen dynamischen Begrenzungs-Set angeben wollen (Spezialparameter-3) TAL wenn Sie die Zählfunktion anwenden wollen SOA wenn Sie die aufsteigende Sortierfunktion anwenden wollen SOD wenn Sie die absteigende Sortierfunktion anwenden wollen NOW wenn Sie bei Sperrsituationen nicht warten wollen Die Einträge RES, LMS, TAL, SOA und SOD sind nur erlaubt bei ...FST und ...SEX. Die Eingabe von RES oder TAL führt dazu, dass die gesamte Treffermenge ermittelt wird. Zum Format der Eingabe siehe Seite 203.
Benutzerinformation:	TAL Rückgabe des Ergebnisses der TALLYING-Funktion im Feld Zähler als 4 byte lange Binärzahl Rückgabe des DATABASE-STATUS
Satzname:	Eingabe eines Satznamens

Setname: CUR.../SEL...
Eingabe eines Setnamens

– Bei CUR... wird die Current Set-Occurrence des angegebenen Sets als Auswahlmenge bestimmt.

– Bei SEL... wird die Set-Occurrence, in der gesucht werden soll, mit Hilfe der SET OCCURRENCE SELECTION bestimmt. Sets mit der SET OCCURRENCE SELECTION-Klausel THRU LOCATION MODE OF OWNER werden genauso behandelt wie Sets mit der SET OCCURRENCE SELECTION-Klausel THRU CURRENT OF SET. Die Wirkung von SEL... entspricht daher der von CUR....

– Bei SYSTEM-Sets kann CUR... verwendet werden, auch wenn es noch keinen CRS gibt.

Realmname: ...SOA/...SOD
Eingabe eines oder mehrerer Feldnamen; die Ergebnismenge eines Suchausdrucks wird nach diesen Feldern sortiert. Die Felder müssen Bestandteil der Satzart sein, die beim Suchausdruck angesprochen wird. Die Sätze werden typgerecht sortiert.

– Bei ...SOA wird aufsteigend sortiert.

– Bei ...SOD wird absteigend sortiert.

Alle angegebenen Felder müssen in der gleichen Richtung sortiert werden.

Feldname: ...ITM/...ITN/...ITP
Eingabe eines oder mehrerer Feldnamen; es wird nach einem Satz gesucht, der in den angegebenen Feldern die gleichen Werte hat, wie sie im Satzbereich vorhanden sind.

Bei ...ITN wird der gemäß Sortierreihenfolge unmittelbar nachfolgende Satz der Set-Occurrence ausgewählt, falls kein Satz mit den im Satzbereich angegebenen Werten gefunden wird. Die Sortierreihenfolge ist festgelegt durch die angegebene Kombination der Feldnamen, wobei diese Kombination von Feldnamen den ASCENDING KEY, DESCENDING KEY oder ein SEARCH KEY USING INDEX des angegeben Set bezeichnen muss. Eine Verletzung dieser Bedingungen quittiert UDS/SQL zur Laufzeit mit DATABASE-STATUS 04101.

Existieren zu einem Schlüsselwert Duplikatsätze, so gibt UDS/SQL von diesen Sätzen den Satz mit der kleinsten Satzfolgenummer (RSQ) aus. Somit können Sie sämtliche Duplikate ermitteln

– mit einem Aufruf FIND7A/FTCH7A ...ITN und einer Folge von Aufrufen FIND4/FTCH4 ... NXT oder

– mit einem Aufruf FIND7A/FTCH7A ...ITN und einer Folge von Aufrufen FIND/FTCH3.

Wenn der im Satzbereich vorgegebene Schlüsselwert größer ist als der größte in der Set-Occurrence vorhandene Schlüsselwert, meldet UDS/SQL den DATABASE-STATUS 04024

Bei ...ITP wird der gemäß Sortierreihenfolge unmittelbar vorhergehende Satz der Set-Occurrence ausgewählt, falls kein Satz mit den im Satzbereich angegebenen Werten gefunden wird. Die Sortierreihenfolge ist festgelegt durch die angegebene Kombination der Feldnamen, wobei diese Kombination von Feldnamen den ASCENDING KEY bzw. DESCENDING KEY des angegeben Set bezeichnen muss. Eine Verletzung dieser Bedingungen quittiert UDS/SQL zur Laufzeit mit DATABASE-STATUS 04101.
Existieren zu einem Schlüsselwert Duplikatsätze, so gibt UDS/SQL von diesen Sätzen den Satz mit der größten Satzfolgenummer (RSQ) aus. Somit können Sie mit einem Aufruf FIND7A/FTCH7A ...ITP und einer Folge von Aufrufen FIND4/FTCH4 ... PRI sämtliche Duplikate ermitteln.
Wenn der im Satzbereich vorgegebene Schlüsselwert kleiner ist als der kleinste in der Set-Occurrence vorhandene Schlüsselwert, meldet UDS/SQL den DATABASE-STATUS 04024.

...SEX
Eingabe eines Suchausdruckes; es werden einer oder mehrere Sätze gesucht, die die Bedingung(en) des Suchausdrucks erfüllen. Der Suchausdruck ist im Anschluss an „Spezialparameter-3“ beschrieben.

Satzbereich: Eingabe der Feldinhalte der angegebenen Felder bei Funktionswahl ...ITM/...ITN/...ITP; sie müssen an den Positionen der Felder im Satzbereich stehen, die dem Subschema-Format des Satzes entsprechen.

FTCH7A
Rückgabe des (ersten) gefundenen Satzes

Spezialparameter-1: Eingabe der Retaining-Parameter bei Zusatzwahl RET

Spezialparameter-2: RES
Eingabe eines Namens für einen Ergebnisset.
RES geben Sie an, wenn UDS/SQL die Treffersätze in einen expliziten dynamischen Set speichern soll, den Sie mit weiteren FIND4/FTCH4-Aufrufen ansprechen können. Auf diese Weise können Sie die Suche nach Sätzen so programmieren, dass insgesamt mehrere Hierarchiestufen der Datenstruktur durchlaufen werden, d.h. für die Suche können Sie über Set-Beziehungen insgesamt Felder verschiedener Satzarten benutzen (komplexe Suchfrage).
Wenn der angegebene Set ein unsortierter dynamischer Set ist, dürfen Sie im Parameter „Realmname“ (s.o.) nicht ...SOA oder ...SOD angeben, da der Durchschnitt einer Treffermenge und eines dynamischen Set nicht sortiert werden kann.
RES dürfen Sie nicht verwenden, wenn Sie im Parameter „Feldname“ ...ITM angegeben haben.

Spezialparameter-3: LMS
Eingabe des Namens für den Set, mit dem die Treffermenge geschnitten werden soll.
LMS dürfen Sie nur angeben, wenn der im Parameter „Setname“ angegebene Set keinen dynamischen Set bezeichnet.
LMS begrenzt die Auswahlmenge auf die Sätze, die

– zur im Parameter „Satzname“ angegebenen Satzart bzw. zur ausgewählten Set-Occurrence des im Parameter „Setname“ angegebenen Set gehören und zusätzlich
– enthalten sind im Set, der im „Spezialparameter-3“ angegeben ist.

Der im „Spezialparameter-3“ angegebene Set darf kein sortierter dynamischer Set sein, da die Durchschnittbildung einer Treffermenge mit einem sortierten dynamischen Set nicht möglich ist.
LMS dürfen Sie nicht verwenden, wenn Sie im Parameter „Feldname“ ...ITM angegeben haben.

```
suchausdruck :: =  { komplex-1[ AN komplex-2] } END
                   { komplex-2                }

komplex-1   ::= bedingung-1[ { AN } bedingung-1]...
                             { OR }

komplex-2   ::= bedingung-2[ AN bedingung-2]...

bedingung-1 ::= n feldname [MSK maske ]rel wert m

bedingung-2 ::= 0 feldname { NXT rel wert } 0
                           { MAX          }
                           { MIN          }
```

Ein Suchausdruck darf insgesamt nicht mehr als 31 Bedingungen enthalten.

Für das logische Bearbeiten des Suchausdrucks und das Auswerten der mehrstufigen Tabellen von ASC-Keys und SEARCH-Keys gelten dieselben Regeln wie bei COBOL-DML.

n
Anzahl linker (öffnender) Klammern *n* = 0 ... 9

Die Anzahl der öffnenden und schließenden Klammern in *komplex-1* muss gleich sein, in *komplex-2* sind keine Klammern gestattet; wegen Formatkompatibilität müssen Sie 0 angeben.

m
Anzahl rechter (schließender) Klammern *m* = 0 ... 9

rel (Relation):

EQU: equal
GTH: greater than
LTH: less than
GEQ: greater than or equal
LEQ: lower than or equal
NEQ: not equal
NGR: not greater than
NLW: not lower than

EQU und NEQ sind in einer *bedingung-2* verboten.
Das Bool'sche NOT ist nicht vorgesehen, kann jedoch durch entgegengesetzte Relationen bewirkt werden.

feldname
Feldname in 8 byte bzw. 30 byte Länge entsprechend der Varianten (CALL8) und (CALL30).

maske
Literal in der Länge des Feldes *feldname*
signifikante Bytes: X'F1'
nicht signifikante Bytes: 'F0'

wert
Literal von gleicher Länge und gleichem Typ wie das Feld *feldname*; es enthält den Vergleichswert.

Um Ausrichtungsprobleme beim Vergleichswert (abhängig von den Konventionen der verwendeten Programmiersprache) zu vermeiden, darf das Literal *wert* mit bis zu sieben Zeichen beginnen, die nicht zum Vergleichswert gehören. Zum Vergleich werden die letzten (rechtsbündigen) Bytes von *wert* in der Länge des Feldes *feldname* herangezogen.

Vergleichen können Sie nur mit Literalen, die in den Suchausdruck eingefügt sind, nicht mit den Inhalten anderer Felder.

MAX/MIN
dienen zum Auffinden des größten und des kleinsten Wertes (in COBOL-DML nicht vorhanden).

END
beendet den Suchausdruck.

Beispiel für Variante (CALL8)

Gesucht wird der Bauer, der die dicksten Kartoffeln erntet; außer Bauern kommen auch Landwirtschaftspolitiker in die Auswahl.

Subschemasatz:

```
01 MITBUERGER.
   02 BERUF         PIC X(12).
   02 ARBEITSGEBIET PIC X(15).
   02 KARTOFFEL     PIC S999.
```

COBOL-DML-Suchausdruck:

```
(BERUF="LANDWIRT" OR (BERUF="ABGEORDNETER" AND
ARBEITSGEBIET="LANDWIRTSCHAFT")) AND KARTOFFEL
NEXT NOT GREATER HIGH VALUE
```

CALL-DML-Suchausdruck:

```
1␣BERUF␣␣␣␣EQU␣LANDWIRT␣␣␣␣␣0␣OR␣
1␣BERUF␣␣␣␣EQU␣ABGEORDNETER␣0␣AN␣
0␣ARBEITSG␣EQU␣LANDWIRTSCHAFT␣␣2␣AN␣
0␣KARTOFFE␣MAX␣0␣END
```

Die Bedingungen folgen lückenlos aufeinander, sie wurden hier nur der Übersichtlichkeit halber zeilenweise geschrieben.

Abschließen der Verarbeitung (FINISC)

Die FINISC-Funktion beendet eine Transaktion. Bisherige Änderungen der Transaktion werden fixiert oder wahlweise zurückgenommen. Gesperrte Realms und Seiten werden freigegeben.

Funktionsname: **FINISC**

Funktionswahl:

CALL-DML	entsprechende COBOL-DML-Anweisung
ALLRLM	FINISH
ALLCAN	FINISH WITH CANCEL

Benutzerinformation: Rückgabe des DATABASE-STATUS

Ausschalten des erweiterten Satzschutzes (FREEC)

Die FREEC-Funktion beendet den KEEP-Status für einen oder mehrere Sätze.

Funktionsname: **FREEC**

Funktionswahl:

CALL-DML	entsprechende COBOL-DML-Anweisung
CORUNT	FREE
ALLREC	FREE ALL

Benutzerinformation: Rückgabe des DATABASE-STATUS

Einen Satz in den Satzbereich transportieren (GETC)

Die GETC-Funktion stellt den Current Record of Rununit (CRU) oder einzelne Felder des CRU im Satzbereich der UWA zur Verfügung.

Funktionsname: **GETC**

Funktionswahl:

CALL-DML	entsprechende COBOL-DML-Anweisung
CORUNT	GET[*satzname*]
ITMNAM	GET *feldname-1*,...

ITMNAM ist nicht erlaubt bei Sätzen mit variablen Feldern.

Zusatzwahl: ITMNAM
Bei Angabe von VAR (wahlfrei) werden die angegebenen Felder in der angegebenen Reihenfolge im Satzbereich abgelegt, unabhängig vom Format des Subschemasatzes und ohne Zwischenräume.

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: Eingabe eines Satznamens

CORUNT
Die Eingabe eines Satznamens ist wahlfrei; der DBH überträgt aber immer den CRU vollständig in den Satzbereich.

Geben Sie einen Satznamen an, so prüft der DBH zunächst, ob der Current of Rununit von dieser Satzart ist, bevor er ihn in den Satzbereich überträgt.

ITMNAM
Der DBH überträgt nur die angegebenen Felder in den Satzbereich; einen Satznamen müssen Sie angeben.

Feldname: ITMNAM
Eingabe eines oder mehrere Feldnamen

Satzbereich:

CORUNT
Rückgabe des Current of Rununit

ITMNAM
Rückgabe einzelner Felder des CRU

Geben Sie die Zusatzwahl VAR nicht an, so werden diese Felder an die Stellen im Satzbereich übertragen, die ihrem Platz im Subschema-Format des Satzes entsprechen.

Geben Sie VAR an, so werden die Inhalte der angegebene Felder lückenlos aneinander gefügt (verkürztes Satzformat).

Abfragen von Datenbankbedingungen (IFC)

Mit der IFC-Funktion können Sie in der Datenbank abfragen,

– ob der Current Record of Rununit (CRU) Owner oder Member ist in einem oder mehreren angegebenen Sets oder

– ob die Current Set-Occurrence Membersätze enthält.

Funktionsname: **IFC**

Funktionswahl:

CALL-DML	entsprechende COBOL-DML-Anweisung
OWNALL	IF OWNER
OWNSET	IF *setname* OWNER
MEMALL	IF MEMBER
MEMSET	IF *setname* MEMBER
TENALL	IF TENANT
TENSET	IF *setname* TENANT
EMPTYS	IF *setname* EMPTY

Das COBOL NOT ist nicht vorgesehen; dies bedeutet jedoch keine tatsächliche Funktionseinschränkung.

Benutzerinformation: Rückgabe des Ergebnisses im Feld Statuscode

000: die Bedingung ist erfüllt

C11: die Bedingung ist nicht erfüllt

Unabhängig vom Ergebnis wird das Programm immer an der angegebenen Rücksprungadresse fortgesetzt (bei der auf den CALL folgenden Anweisung bzw. Reg.14).

Rückgabe des DATABASE-STATUS

Setname: ...SET/EMPTYS
Eingabe eines Setnamens; nur der angegebene Set wird geprüft.

Einschalten des erweiterten Satzschutzes (KEEPC)

Die KEEPC-Funktion entzieht den Current of Rununit solange dem Zugriff anderer Transaktionen, bis Sie den Satzschutz mit FREEC aufheben oder mit FINISC die Transaktion beenden.

Funktionsname: **KEEPC**

Funktionswahl entfällt

Benutzerinformation: Rückgabe des DATABASE-STATUS

Ändern bereits gespeicherter Sätze (MODIF1/2)

Die MODIF1/2-Funktionen

- ersetzen die Werte aller oder der angegebenen Felder des CRU durch Werte aus dem Satzbereich,
- ändern die Position des CRU innerhalb der Set-Occurrence in Übereinstimmung mit der Set-Order, wenn ein Sortierschlüssel geändert wird,
- hängen den CRU wahlweise aus seiner derzeitigen Set-Occurrence aus und in eine neue ein, die mit Hilfe den Set-Auswahlmechanismen bestimmt wird.

Sie können wahlweise das Verändern aller oder einzelner betroffener Currency-Informationen für Sets unterdrücken.

Die MODIF1/2-Funktionen können Sie mit 2 verschiedenen Funktionsnamen ansprechen.

Ändern des ganzen Satzes

Funktionsname: **MODIF1**

Funktionswahl:

CALL-DML	entsprechende COBOL-DML-Anweisung
CORUNT	MODIFY *satzname*
INCALL	MODIFY *satzname* INCLUDING ALL MEMBERSHIP
ONLALL	MODIFY *satzname* ONLY ALL MEMBERSHIP
INCSET	MODIFY *satzname* INCLUDING *setname-1*,... MEMBERSHIP
ONLSET	MODIFY *satzname* ONLY *setname-1*,... MEMBERSHIP

Zusatzwahl: Eingabe von RET (wahlfrei), NOW (wahlfrei);
Retaining ist nur für Sets möglich.

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: Eingabe des Satznamens des Current of Rununit

Setname: ...SET
Eingabe eines oder mehrerer Setnamen; die Mitgliedschaft des CRU in den angegebenen Sets wird geprüft und, soweit dies erforderlich und ausschließlich über Currency-Angaben möglich ist, geändert.

Wenn die Mitgliedschaft des CRU in einem Set geändert wird, der mit der SET OCCURRENCE SELECTION-Klausel THRU LOCATION MODE OF OWNER definiert ist, geht MODIF1 genauso vor wie bei der SET OCCURRENCE SELECTION-Klausel THRU CURRENT OF SET.

Satzbereich: CORUNT/INCALL/INCSET
Sie müssen den Satz im vollständigen Subschema-Format im Satzbereich übergeben. Alle Felder des Satzes, die im Subschema angegeben sind, werden mit Inhalten aus dem Satzbereich überschrieben.

ONLALL/ONLSET
Der Satzbereich wird nicht verwendet; Feldinhalte bleiben unverändert.

Spezialparameter-1: Eingabe von Retaining-Parametern bei Zusatzwahl RET; Retaining ist nur für Sets möglich, da die CRS geändert werden.

Ändern einzelner Felder

Funktionsname: **MODIF2**

Diese Funktion ist nicht erlaubt bei Sätzen mit variablen Feldern.

Funktionswahl:

CALL-DML	entsprechende COBOL-DML-Anweisung
CORUNT	MODIFY *feldname*,...
INCALL	MODIFY *feldname*,... INCLUDING ALL MEMBERSHIP
INCSET	MODIFY *feldname*,... INCLUDING *setname-1*,... MEMBERSHIP

Zusatzwahl: Eingabe von:
RET (wahlfrei); Retaining nur für Sets
VAR (wahlfrei); verkürztes Satzformat

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: Eingabe des Satznamens des Current of Rununit

Setname: INCSET
Eingabe eines oder mehrerer Setnamen; es gilt das bei MODIF1 Gesagte

Feldname: Eingabe eines oder mehrerer Feldnamen; im Satz werden nur die Inhalte der angegebenen Felder geändert, die der übrigen Feldinhalte bleiben erhalten.

Satzbereich: Eingabe der zu ändernden Feldinhalte

Geben Sie die Zusatzwahl VAR nicht an, so müssen die Inhalte an den Stellen im Satzbereich stehen, die der Position der Felder im Subschema-Format des Satzes entsprechen.
Geben Sie VAR an, so müssen die Feldinhalte in der Reihenfolge der Feldnamen lückenlos linksbündig (am Beginn des Parameterfeldes „Satzbereich“ anfangend) aufeinander folgen.

Spezialparameter-1: siehe MODIF1 auf Seite 252

Eröffnen der Verarbeitung (READYC)

Die READYC-Funktion eröffnet eine Transaktion/Verarbeitungskette und bereitet einen oder mehrere Realms für die Verarbeitung vor.

Wahlweise können Sie verlangen, dass beim Eröffnen einer Transaktion nicht gewartet, sondern die Kontrolle an das Anwenderprogramm zurückgegeben wird (DATABASE-STATUS 12099), falls ein Realm von einer anderen Transaktion gesperrt ist.

Funktionsname: **READYC**

Funktionswahl:

CALL-DML	entsprechende COBOL-DML-Anweisung
ALL...	alle Realms des Subschemas eröffnen
RLM...	nur die angegebenen Realms eröffnen
...RTR	USAGE MODE IS RETRIEVAL
...UPD	USAGE MODE IS UPDATE
...PRT	USAGE MODE IS PROTECTED RETRIEVAL
...PUP	USAGE MODE IS PROTECTED UPDATE
...ERT	USAGE MODE IS EXCLUSIVE RETRIEVAL
...EUP	USAGE MODE IS EXCLUSIVE UPDATE

Alle 12 möglichen Kombinationen sind zugelassen.

Zusatzwahl: Eingabe von: NOW (wahlfrei)

Benutzerinformation: Rückgabe des DATABASE-STATUS
Rückgabe des DB-Kennzeichens

Das DB-Kennzeichen muss bei allen folgenden DML-Aufrufen der Verarbeitungskette wieder angegeben werden.

Realmname: ...RLM
Eingabe von einem oder mehreren Realm-Namen;

Spezialparameter-1: Eingabe des Subschemanamens in voller Länge (30 Bytes); ist der Name kürzer, müssen Sie mit Leerzeichen auffüllen.
Die CALL-DML gestattet es, vom selben Modul aus verschiedene Subschemata aufzurufen; daher müssen Sie den Subschemanamen beim READYC-Aufruf mit übergeben. Die zugehörigen SSITAB-Module müssen dem CALL-DML-Umsetzer zur Verfügung stehen.

Spezialparameter-2: Wird nicht mehr benötigt; evtl. vorhandene Angaben werden von UDS/SQL ignoriert.
Die PRIVACY-Daten werden außerhalb des Programms durch openUTM bzw. BS2000 an UDS/SQL übergeben.

Speichern von Sätzen (STORE1/2, STOR1L/2L)

Die Funktionen STORE1/2 und STOR1L/2L speichern einen Satz in die Datenbank und fügen ihn in alle Sets ein, in denen seine Satzart als AUTOMATIC Member definiert ist. Der neue Satz wird dadurch zum

- Current of Rununit,
- Current of Realm des Realm, in dem er gespeichert ist,
- Current of Record seiner Satzart,
- Current of Set aller Sets, in denen seine Satzart als Ownersatzart oder AUTOMATIC Membersatzart definiert ist.

Wahlweise können Sie alle oder einige von dem STORE1/2- bzw. STOR1L/2L-Aufruf betroffene Currency-Informationen erhalten außer der des CRU.

Die Funktionen STORE1/2 und STOR1L/2L behandeln Sets, die mit der SET OCCURRENCE SELECTION-Klausel THRU LOCATION MODE OF OWNER definiert sind, genauso wie Sets, die mit der SET OCCURRENCE SELECTION-Klausel THRU CURRENT OF SET definiert sind.

STORE1/2 und STOR1L/2L unterscheiden sich hinsichtlich des von Ihnen anzugebenden Database-Key-Wertes.

STOR1L und STOR2L können nur in Verbindung mit einem SSITAB-Modul einer Version ab UDS/SQL V2.0 ausgeführt werden (siehe Handbuch „Aufbauen und Umstrukturieren“). Andernfalls meldet UDS/SQL den Statuscode C98.

Speichern von vollständigen Sätzen

Funktionsname: **STORE1**

Funktionsname: **STOR1L**

Funktionswahl:

CALL-DML	entsprechende COBOL-DML-Anweisung
RECNAM	STORE *satzname*
IMPDAT	STORE *satzname*

IMPDAT müssen Sie verwenden, wenn LOCATION MODE DIRECT oder LOCATION MODE DIRECT-LONG definiert ist oder mehrere Realms in der WITHIN-Klausel angegeben sind, wobei die Satzart nicht Membersatzart einer verteilbaren Liste ist. Diese implizit definierten Daten (Database-Key-Wert bzw. Realm-Name) müssen Sie im Spezialparameter-2 übergeben.

Zusatzwahl: Eingabe von RET (wahlfrei)

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: Eingabe eines Satznamens

Satzbereich: Eingabe des abzuspeichernden Satzes im Subschema-Format. Bei Sätzen mit variablen Feldern müssen Sie das Längenfeld des variablen Feldes mit einer Binärzahl versorgen. Der Satz wird nur in entsprechender Länge gespeichert.

Spezialparameter-1: Eingabe von Retaining-Parametern bei Zusatzwahl RET

Spezialparameter-2: IMPDAT
Eingabe eines Database-Key-Wertes (bei LOCATION MODE DIRECT bzw. LOCATION MODE DIRECT-LONG) und/oder eines Realm-Namens, wenn mehrere Realm-Namen in der WITHIN-Klausel angegeben sind (Näheres siehe Seite 210).
Wenn Sie im Fall LOCATION MODE DIRECT-LONG einen Database-Key-Wert mit einer REC-REF > 254 und/oder einer RSQ > 2^{24}-1 angeben wollen, müssen Sie STOR1L verwenden.

Speichern von einzelnen Feldern oder komprimierten Sätzen

Funktionsname: **STORE2**

Funktionsname: **STOR2L**

Funktionswahl:

CALL-DML	entsprechende COBOL-DML-Anweisung
ITMNAM	speichern der angegebenen Felder
IMPDAT	Abspeichern der angegenen Felder und Übergabe implizit definierter Daten

Dieses Format ist eine Funktionserweiterung gegenüber der COBOL-DML. Es gestattet das Arbeiten mit einem verkürzten Satzformat. Dieses Format ist nicht erlaubt bei Sätzen mit variablem Feld.

Zusatzwahl: Eingabe von:
RET (wahlfrei) für Retaining
VAR (wahlfrei) für verkürztes Satzformat

Benutzerinformation: Rückgabe des DATABASE-STATUS

Satzname: Eingabe eines Satznamens

Feldname: Eingabe eines oder mehrerer Feldnamen.
Felder deren Namen nicht angegeben sind, werden von UDS/SQL mit binär Null versorgt. Wenn die SSL-Definition für die bei „Satzname“ angegebene Satzart die Klausel „COMPRESSION FOR ALL ITEMS“ enthält, speichert UDS/SQL die mit binär Null belegten Felder nicht in der Datenbank ab.

Satzbereich: Eingabe der Feldinhalte der angegebenen Felder.
Verwenden Sie VAR nicht, so müssen die Feldinhalte an den Stellen des Satzbereichs stehen, die ihrer Position im Format des Subschemasatzes entsprechen.
Geben Sie VAR an, so müssen die Feldinhalte lückenlos linksbündig in der angegebenen Reihenfolge aufeinander folgen. Diese Angabe führt zusammen mit einer SSL-Definition COMPRESSION FOR ALL ITEMS für die betroffene Satzart zur komprimierten Speicherung des Satzes.

Spezialparameter-1: Eingabe von Retaining-Parametern bei Zusatzwahl RET.

Spezialparameter-2: wie bei STORE1, STOR1L

8.4 CALL-DML-Assembler-Makros

Zur Unterstützung von UDS/SQL-Anwendern, die von Assembler-Programmen aus mit CALL-DML arbeiten, stehen für die Variante (CALL8) folgende Makros zur Verfügung:

Makro	Funktion		Anwendung
	statisch zur Übersetzungszeit	**dynamisch zur Ablaufzeit**	
DSCAL	Implizite Parameterleiste (im Befehlscode) generieren und vorbesetzen	– ggf. implizite Parameterleiste vervollständigen – CALL-DML-Aufruf mit der impliziten Parameterleiste durchführen	CALL-DML-Aufruf mit impliziter Parameterleiste und komfortablem Parametermechanismus Alternative: DSCAP + DSCDF
DSCAP	-	– explizit definierte Parameterleiste übernehmen – ggf. explizit definiert Parameterleiste füllen – CALL-DML-Aufruf mit der expliziten Parameterleiste durchführen	CALL-DML-Aufruf mit expliziter Parameterleiste und komfortablem Parametermechanismus (wahlfrei)
DSCDF	Parameterleiste (explizit) in der aktuellen CSECT bzw. DSECT generieren und vorbesetzen	-	CALL-DML-Parameterleiste (explizit) generieren
DSCPA	Benutzerinformationsbereich und/oder vordefinierte CALL-DML-Parameter-Konstanten generieren	-	Unterstützung der ASSEMBLER-Programmierung an der CALL-DML-Daten-Schnittstelle

Tabelle 27: CALL-DML-Assembler-Makros

Parameter-Mechanismus

In den Makros DSCAL, DSCAP und DSCDF werden die CALL-DML-Parameter mit den jeweils ersten zwölf Stellungsparametern dieser Makros angesprochen und dabei symbolisch mit FCOD, FOPT, SOPT, UINF, RECN, SETN, RLMN, ITMN, RECA, SPP1, SPP2, SPP3 bezeichnet (siehe Abschnitt „Parameterdefinitionen“ auf Seite 198).

Für Parameter, die die Adresse eines Datenbereichs angeben, sind die folgenden drei Darstellungsformen möglich:

1. *symbol*
   Die Adresse des Datenbereichs wird durch den symbolischen Ausdruck *symbol* definiert. Sie muss als A-Konstante darstellbar sein, d.h. der Datenbereich darf nicht in einer DSECT liegen, muss aber nicht innerhalb der aktuellen Basisadressierung liegen.

   Beispiel für „symbol“

   ```
   RECORD + L'ITEM-2
   ```

2. **symbol*
   Die Adresse des Datenbereichs wird durch den symbolischen Ausdruck **symbol* definiert. Sie muss als S-Konstante darstellbar sein, d.h. der Datenbereich muss entweder in einer (adressierten) DSECT liegen oder innerhalb der aktuellen Basisadressierung.

   *Beispiel für „*symbol“*

   ```
   *BUFFER + 16
   ```

3. (*r*)
   Die Adresse des Datenbereiches steht im Register (r).

   Beispiel für „(r)“:

   ```
   (3) oder (R12) oder (WORKREG), wobei
   R12      EQU 12
   WORKREG  EQU 7 ist
   ```

Welche dieser drei Darstellungsformen im Einzelfall zulässig sind, wird bei der Beschreibung der Makroparameter angegeben.

DSCAL

Der Makro DSCAL erzeugt einen CALL-DML-Aufruf mit einer impliziten Parameterleiste, d.h. Sie müssen sich um Definition und Versorgung der Parameterleiste nicht selbst kümmern.

```
[name] DSCAL fcod,fopt,...,spp3
```

name gibt dem Makroaufruf einen symbolischen Namen (wahlfrei)

fcod,fopt,...spp3
entsprechen den 12 CALL-DML-Parametern (siehe Abschnitt „Parameterdefinitionen“ auf Seite 198); für jeden dieser Parameter sind die Darstellungsformen *symbol*, **symbol* und (*r*) zulässig. Die Regeln für die Parameterdefinition (siehe Seite 199) müssen Sie genau beachten. Weglassen dürfen Sie nur solche Parameter, die nicht verwendet werden und am Ende stehen.

Beispiel

```
CONN1    DSCAL CONNEC,TOALL,RTSHORT,(RUSINF),*LEER8,NIL,NIL,NIL,NIL,    -
               RETSET
```

DSCAP

Der Makro DSCAP erzeugt einen CALL-DML-Aufruf mit einer expliziten Parameterleiste, d.h. Sie können die Parameterleiste selbst definieren und versorgen.

DSCAP müssen Sie an Stelle von DSCAL in folgenden Fällen verwenden:

– Pro Transaktion soll nur eine Parameterleiste existieren.

– Die Parameterleiste soll nicht im Befehlscode stehen (sondern z.B. in nur temporär vorhandenen Speicherbereichen).

– Das Anwendungsprogramm soll die Parameterleiste vollständig selbst versorgen.

```
[name] DSCAP [fcod,fopt,...,spp3][,PARAM=param]
```

---

name gibt dem Makroaufruf einen symbolischen Namen (wahlfrei)

fcod,*fopt*,...*spp3*
Angabe wie bei DSCAL (wahlfrei)

Haben Sie keine Parameter *fcod*,*fopt*,... angegeben, so müssen Sie die Parameterleiste selbst definieren (siehe Makro DSCDF auf Seite 263) und mit den Adressen der benötigten Datenbereiche füllen.

param ermöglicht es Ihnen, die Adresse einer Parameterleiste in den Darstellungsformen *symbol*, **symbol* oder (*r*) anzugeben. Fehlt dieser Parameter, so wird die Adresse der Parameterleiste in Register 1 erwartet.

Beispiele

```
a)  DSCAP PARAM=*DMLP0001
b)  DSCAP PARAM=(R7)
c)  L     1,PARADDR
    DSCAP
d)  DSCAP CONNEC,TOALL,RTSHORT,(RUSINF),*LEER8,NIL,NIL,NIL,NIL,   -
          RETSET,PARAM=PARLEIST
```

DSCDF

Der Makro DSCDF generiert eine Parameterleiste für CALL-DML-Aufrufe mit dem Makro DSCAP und besetzt sie (statisch) vor.

```
[name] DSCDF fcod,fopt,...,spp3[,SUFFIX=x]
```

---

name gibt der generierten Parameterleiste einen symbolischen Namen (wahlfrei)

fcod,fopt,...,spp3
entsprechen den 12 CALL-DML-Parametern (siehe Seite 197). Für jeden dieser Parameter ist nur die Darstellungsform *symbol* zulässig. Sie können einzelne oder auch alle Parameter weglassen; die entsprechenden Werte der Parameterleiste werden dann mit binär Null vorbesetzt.

x dient dazu, Namenskollisionen zwischen verschiedenen Parameterleisten zu vermeiden (wahlfrei).

Ein DSCDF-Aufruf generiert an der Stelle, an der er in der aktuellen CSECT bzw. DSECT steht, folgenden Code:

```
         DS    0F
DMLPx    DS    0CL48
[Name    EQU   DMLPx]
FCODx    DC    A(symbol bzw. 0)     Funktionsname
FOPTx    DC    A(       "     )     Funktionswahl
SOPTx    DC    A(       "     )     Zusatzwahl
UINFx    DC    A(       "     )     Benutzerinformation
RECNx    DC    A(       "     )     Satzname
SETNx    DC    A(       "     )     Setname
RLMNx    DC    A(       "     )     Realmname
ITMNx    DC    A(       "     )     Feldname
RECAx    DC    A(       "     )     Satzbereich
SPE1x    DC    A(       "     )     Spezialparameter-1
SPE2x    DC    A(       "     )     Spezialparameter-2
SPE3x    DC    A(       "     )     Spezialparameter-3
```

Für *x* werden die ersten vier Zeichen des Parameters SUFFIX eingesetzt.

Beispiel

Makroaufruf:

```
CONNPAR  DSCDF CONNEC,TOALL,RTSHORT,USINFO,BLANK8,NIL,NIL,NIL,NIL,      -
               RETSET,SUFFIX=CONN
```

Generierte Parameterleiste:

```
         DS    0F
DMLPCONN DS    0CL48
CONNPAR  EQU   DMLPCONN
FCODCONN DC    A(CONNEC)
FOPTCONN DC    A(TOALL)
SOPTCONN DC    A(RTSHORT)
UINFCONN DC    A(USINFO)
RECNCONN DC    A(BLANK8)
SETNCONN DC    A(NIL)
RLMNCONN DC    A(NIL)
ITMNCONN DC    A(NIL)
RECACONN DC    A(NIL)
SPE1CONN DC    A(RETSET)
SPE2CONN DC    A(0)
SPE3CONN DC    A(0)
```

DSCPA

Der Makro DSCPA unterstützt den Anwender an der Datenschnittstelle von CALL-DML.

```
[name] DSCPA [option]
```

name gibt dem Makroaufruf einen symbolischen Namen (wahlfrei); wird bei OPTION DMLPAR mit dem Anfang der generierten Konstanten gleichgesetzt.

option hierfür können Sie Folgendes angeben (wahlfrei):

USERINF
In der aktuellen CSECT bzw. DSECT wird die Definition des Bereichs Benutzerinformation generiert. Diese Bereichsdefinition sollten Sie auf Wortgrenze ausrichten.

USERINFDS
Die Definition des Bereichs Benutzerinformation wird als DSECT USERINF generiert.

DMLPAR
Als symbolisch ansprechbare Konstanten werden in der aktuellen CSECT bzw. DSECT generiert:

- alle CALL-DML-Funktionsnamen
- alle Funktionswahl-Parameter aller Funktionsnamen (außer LOOKC)
- die wichtigsten Zusatzwahl-Parameter

DMLPARDS
Generiert eine DSECT DMLPAR mit den unter DMLPAR genannten, symbolisch ansprechbaren Konstanten, sodass Sie auch von anderen Modulen symbolisch auf die Konstanten zugreifen können.

Ein **fehlender** Parameter OPTION bewirkt dasselbe wie DSCPA USERINFDS und DSCPA DMLPAR zusammen.

Die generierten symbolischen Namen und Werte können Sie einer Auflistung des DSCPA-Makros oder entsprechenden Probe-Aufrufen entnehmen.

Beispiele

```
          DSCPA USERINFDS
CDMLCON   DSCPA DMLPAR
```

8.5 LOOKC-Funktion

Sie können sich mit der LOOKC-Funktion Strukturinformationen aus dem Subschema holen. Folgende Strukturinformationen sind erhältlich:

vorgegeben	Strukturinformationen über
SET	Owner- und Membersätze des Set
SATZ	Sets, in denen der Satz Owner oder Member ist
SATZ	Felder des Satzes
FELDSTRUKTUR	Felder der Struktur
SATZ	Bereiche, in denen Ausprägungen des Satzes stehen können
SET	Schlüssel des Set
SCHLÜSSEL	Felder im Schlüssel
FELD	Schlüssel, in denen das Feld enthalten ist
SET UND FELD	Schlüssel der Sets, in denen das Feld enthalten ist

Tabelle 28: Strukturinformationen

Eine Klassifizierung der LOOKC-Funktionen nach Suchbegriff und Inhalt der Antwort ergibt die folgenden beiden Funktionsgruppen:

1. LOOKC nach einem Namen (Funktionswahl:'...NAM')

   Der Suchbegriff besteht aus dem vollen Namen (Länge: 30 byte) und dem Typ (Realm, Satz usw.) des Elements. Die Antwort setzt sich zusammen aus einem kurzen Verweis (Reference), der das Element intern identifiziert, aus dem Typ selbst und aus zusätzlichen Informationen.

2. LOOKC nach einem Typ (Funktionswahl:'...RLM','...REC','...SET','...ITM','. ..KEY')

   Es kann sich um einen Realm-, Satz-, Set-, Feld- oder Schlüsseltyp handeln. Suchargument ist die interne Reference, in einigen Fällen auch zwei oder mehr, und die spezielle Beschreibung (siehe Seite 270).

   Beispiel

   Sie suchen den Schlüssel eines Set:
   Dazu brauchen Sie die Setreference, möglicherweise die Satzreference und die Lage des Feldes im Satz, das zum Schlüssel gehört.

Eine weitere Unterscheidung der LOOKC-Funktionen können Sie vornehmen, je nachdem, ob durch den LOOKC-Aufruf eine einzelne Antwort (ein LOOKC-Block) oder ein Ergebnisvektor, dessen Komponenten einzelne LOOKC-Blöcke sind, geliefert wird. Im zweiten Fall spezifizieren Sie im Spezial-Parameter-1 (10. SPP1) eine sog. LOOKC-Zahl (2 byte lang, binär codiert, zwischen den Grenzen 1 und 255 einschließlich). Sie gibt an, wie viele

LOOKC-Blöcke maximal in den Satzbereich hintereinander zu liegen kommen sollen. Die Länge eines LOOKC-Blockes beträgt 56 byte. Folgende Bedingung müssen Sie dabei beachten:

Länge des Satzbereichs ≥ LOOKC-Zahl * 56

Einfache LOOKC-Funktionen

- Funktionswahl ’SPC...’ und ’FST...’ (SPECIFIED, FIRST)

  Der Benutzer muss einen LOOKC-Block nach den Vorschriften der Schnittstellen-Tabelle mit Suchbegriffen initiieren. (Suchbegriffe sind z.B. für den Wahleintrag ’SPCNAM’ Name und Typ, für den Wahleintrag ’FSTNAM’ nur der Typ.)

- Funktionswahl ’NXT...’ (NEXT)

  Das Ergebnis eines LOOKC-Aufrufs kann als Eingabe für einen weiteren Aufruf mit dem NEXT-Wahleintrag dienen. Zwischen den beiden Aufrufen muss jedoch der LOOKC-Block (eventuell durch Sicherstellen) gegen Überschreiben geschützt werden.

Zusammengesetzte LOOKC-Funktionen

- Funktionswahl ’ALL...’

  Ein LOOKC-Block zu Beginn des Satzbereichs muss initiiert werden. Die Antwort besteht aus so vielen LOOKC-Blöcken, wie vom System gefüllt werden können (entsprechend dem Inhalt an Subschema-Daten in der Datenbank). Der Benutzer kann die Anzahl der LOOKC-Blöcke je Aufruf durch die LOOKC-Anzahl begrenzen.

- Funktionswahl ’FRT...’ (FROM-TO)

  Hier hat der Anwender die Möglichkeit, durch Initiieren zweier LOOKC-Blöcke die Grenzen einer Antwortmenge zu bestimmen. Die Antwort umfasst alle Elemente, deren Wert (z.B. Namen) nicht kleiner als der im ersten LOOKC-Block angegebene Suchbegriff und nicht größer als der im zweiten LOOKC-Block angegebene Suchbegriff ist. Durch die Antwort werden die LOOKC-Blöcke, die die Suchbegriffe enthalten, überschrieben.

- Funktionswahl ’LIS...’ (LIST)

  Dieser Wahleintrag entspricht der Funktionswahl ’SPC...’ bis auf die Tatsache, dass hier so viele LOOKC-Blöcke zu initiieren sind, wie man durch die LOOKC-Zahl angibt. Die Antworten werden genau in die Blöcke zurückgeschrieben, in die die entsprechenden Suchbegriffe zuvor eingetragen wurden.

- Funktionswahl ’OM-NAM’ (OWNER-MEMBER) und Zusatzwahl: SET...’

  Bei diesem speziellen LOOKC-Aufruf wird ein LOOKC-Block mit dem gewünschten Set-Verweis als Suchbegriff initiiert. Die Antwort besteht aus je einem LOOKC-Block für Owner und Member des spezifizierten Sets; d.h. die LOOKC-Zahl muss mit dem Wert ’2’ initiiert worden sein.
  Im Falle von SYSTEM-Sets oder Dynamic-Sets wird im LOOKC-Block des Owners lediglich der externe Name ’SYSTEM␣␣␣(30)’ bzw. ’DYNAMIC␣␣␣(30)’ eingesetzt.

- Zusatzwahl NXA.../...NXA: NEXT PART OF ANSWER

  Enthält bei einer zusammengesetzten LOOKC-Funktion der Ergebnisvektor mehr Komponenten, als durch die LOOKC-Zahl abgedeckt ist, können Sie die vorangegangene LOOKC-Funktion mit der Zusatzwahl NXA... bzw. ...NXA (je nach Funktionswahl) erneut aufrufen, um so den nächsten Teil des Ergebnisvektors abzurufen. Diesen Schritt können Sie so oft wiederholen, bis der ganze Ergebnisvektor ausgegeben ist. Achten Sie darauf, dass zwischen den einzelnen Aufrufen einer solchen Sequenz keine anderen CALL-DML-Aufrufe erfolgen.

8.5.1 Der LOOKC-Block

Die Schnittstelle für den Austausch von Ein-/Ausgabedaten bei einem LOOKC-Aufruf heißt **LOOKC-Block** und ist dem Satzbereich (RECA) überlagert. Die Länge eines LOOKC-Blocks beträgt 56 byte. Der LOOKC-Block enthält:

– eine generelle Beschreibung für alle LOOKC-Aufrufe (Länge: 38 byte)
– eine spezielle Beschreibung für die einzelnen LOOKC-Aufrufe (Länge: 18 byte)

Die Werte in der Spalte „Inhalt“ der nachfolgenden Tabellen zur Beschreibung des LOOKC-Blocks sind wie folgt zu lesen:

– Bei Angabe „(Byte)“ ist das vollständige Bitmuster (d.h. der Wert) des betreffenden Bytes im LOOKC-Block auf den in der Spalte „Inhalt“ angegebenen Wert zu prüfen.

– Bei Angabe „(Bit)“ ist innerhalb des betreffenden Bytes des LOOKC-Blocks nur das Bit mit der angegebenen Wertigkeit relevant und auf „Bit gesetzt“ zu prüfen. So ist z.B. bei der Angabe 01 in der Spalte „Inhalt“ das letzte Bit des betreffenden Bytes auszuwerten, bei Angabe 08 ist das 5. Bit (von links) auszuwerten usw.

Die bei den verschiedenen LOOKC-Aufrufen notwendigen Eingabedaten und zurückgelieferten Ausgabedaten finden Sie in der Übersicht im Abschnitt „LOOKC-Tabellen“ auf Seite 278.

Generelle Beschreibung für alle LOOKC-Aufrufe

Bedeutung	Inhalt	Länge	Distanz	Typ
Name des Elements		30[1]	0	Character
Reference 1		2	30	binär
Reference 2		2	32	binär
Datentyp – ohne Angabe – Realm – Satzart – Set – Feld – Key	(Byte) 00 01 02 03 04 05	1	34	binär
Mehrdeutigkeit – Name ist eindeutig – Name ist mehrfach vorhanden	(Bit) 00 40	1	35	binär
Ergebnis – in Ordnung – nicht gefunden – kein weiteres Element gefunden	(Bit) 00 01/02/08 04	1	36	binär
Reserve		1	37	

Tabelle 29: Generelle Beschreibung im LOOKC-Block

1 auch bei Variante (CALL8)

Spezielle Beschreibung

Die 18 byte lange spezielle Beschreibung ist unterschiedlich aufgebaut, je nachdem, ob Angaben zu einem Realm, Satz, Set, Feld oder Schlüssel gemacht werden.

Aus Kompatibilitätsgründen sind in der speziellen Beschreibung des LOOKC-Blocks folgende Bereiche jeweils in einer kurzen und in einer langen Variante vorhanden:

- Bereiche „kurze/lange Angabe zur Lage im Satzbereich“ (Displacement innerhalb des COBOL-BIB) in der speziellen Beschreibung zu Satzart (siehe Seite 271).

  UDS/SQL legt die Information „Displacement innerhalb des COBOL-BIB“ wie folgt ab:

  – Wenn das Displacement $\leq 2^{16}$-1 ist, legt UDS/SQL das Displacement sowohl im kurzen als auch im langen Bereich ab.

  – Wenn das Displacement $\geq 2^{16}$ ist, legt UDS/SQL das Displacement im langen Bereich ab und schreibt in den kurzen Bereich den Wert X‘FFFF‘ (ungültiger Wert für das Displacement).

- Bereiche (kurz/lang) für Satzverweise und Bereiche (kurz/lang) für Setverweise in den speziellen Beschreibungen zu Set (siehe Seite 272), Feld (siehe Seite 273) und Schlüssel (siehe Seite 274).

  Für die Belegung der Bereiche für Satz- oder Setverweise gilt im Einzelnen:

  – bei der Eingabe:
    Eine Satzart-/Setnummer $\leq$ 254 geben Sie wahlweise im entsprechenden kurzen oder langen Bereich an. Wenn Sie die Nummer im kurzen Bereich angeben, ist der Inhalt des langen Bereichs nicht relevant. Wenn Sie im kurzen Bereich binär 0 angeben, so erwartet UDS/SQL die Nummer im langen Bereich.
    Eine Satzart-/Setnummer > 254 müssen Sie im entsprechenden langen Bereich angeben. Den entsprechenden kurzen Bereich versorgen Sie mit dem (ungültigen) Wert X‘00‘.

  – bei der Ausgabe:
    Eine Satzart-/Setnummer $\leq$ 254 legt UDS/SQL sowohl im entsprechenden kurzen als auch im entsprechenden langen Bereich ab.
    Eine Satzart-/Setnummer > 254 legt UDS/SQL im entsprechenden langen Bereich ab. In den entsprechenden kurzen Bereich schreibt UDS/SQL den Wert X‘00‘(ungültige Satzart-/Setnummer).

Spezielle Beschreibung zu Realm im LOOKC-Block

Bedeutung	Inhalt	Länge	Distanz	Typ
Filler		4	0	binär
Realmzustand – nicht temporär – temporär	(Bit) ungleich 80 80	1	4	binär
Reserve		13	5	

Tabelle 30: Spezielle Beschreibung zu Realm im LOOKC-Block

Spezielle Beschreibung zu Satzart im LOOKC-Block

Bedeutung	Inhalt	Länge	Distanz	Typ
lange Angabe zur Lage im Satzbereich (Displacement innerhalb des COBOL-BIB)		4	0	binär
Länge der Satzart		2	4	binär
LOCATION MODE – DIRECT – CALC DUPLICATES – CALC NO DUPLICATES – kein LOCATION MODE	(Byte) 00 01 02 03	1	6	binär
Details[1] – Satzart komprimiert – Satzart mit SEARCH-Key – Satzart in verschiedenen Realms	(Bit) 80 20 10	1	7	binär
kurze Angabe zur Lage im Satzbereich (Displacement innerhalb des COBOL-BIB)		2	8	binär
Reserve		8	10	

Tabelle 31: Spezielle Beschreibung zu Satzart im LOOKC-Block

1 Hier sind Kombinationen möglich

Spezielle Beschreibung zu Set im LOOKC-Block

Bedeutung	Inhalt	Länge	Distanz	Typ
Ownersatzverweis (lang)		2	0	binär
Membersatzverweis (lang)		2	2	binär
Ownersatzverweis (kurz)		1	4	binär
Membersatzverweis (kurz)		1	5	binär
Set-Order – SORTED – FIRST – LAST – NEXT – PRIOR – SORTED INDEXED	(Byte) 00 11 22 44 78 80	1	6	binär
CONNECT-Typ – AUTOMATIC – MANUAL	(Byte) 00 01	1	7	binär
DISCONNECT-Typ – MANDATORY – OPTIONAL	(Byte) 00 01	1	8	binär
SET SELECTION – THRU OWNER – THRU CURRENT	(Byte) 00 01	1	9	binär
Spezielle Typen[1] – Singulärer Set – DYNAMIC SET – impliziter Set	(Bit) 80 40 20	1	10	binär
Reserve		7	11	

Tabelle 32: Spezielle Beschreibung zu Set im LOOKC-Block

1 Hier sind Kombinationen möglich.
Falls die entsprechenden Bits nicht gesetzt sind: kein Singulärer Set / DYNAMIC SET / impliziter Set

Spezielle Beschreibung zu Feld im LOOKC-Block

Bedeutung	Inhalt	Länge	Distanz	Typ
Filler		2	0	binär
Verbindung Schlüsselfeld-Set: Setverweis (lang) auf den Set, der zum bei Distanz 14 referenzierten Schlüssel gehört.		2	2	binär
Stufennummer		1	4	binär
nächste Stufennummer		1	5	binär
Feldtyp – Database Key – dezimal gepackt – binär – Zeichen – entpackt mit Vorzeichen – entpackt ohne Vorzeichen – Datengruppe – national – nationale Datengruppe	(Byte) 00 01 02 04 05 06 0F 14 1F	1	6	binär
Skalenfaktor Bit 0 = Vorzeichen Bit 1-7 = Wert		1	7	binär
Feldlänge		2	8	binär
Anzahl der Occurrences		2	10	binär
LOCATION MODE IS CALC-Anzeige – einzelner Schlüssel – zusammengesetzter Schlüssel – kein Schlüssel	(Bit) 40 20 00	1	12	binär
SEARCH KEY...USING CALC /SEARCH KEY...USING INDEX- / Sort-Key -Anzeige[1] – kein Schlüssel – einzelner Schlüssel – zusammengesetzter Schlüssel – Feld in mehreren Schlüsseln verwendet – Wiederholungsgruppenfeld – Feld ist eine Datengruppe, die einen zusammengesetzten Schlüssel bildet	(Bit) 80 40 20 10 01 08	1	13	binär

Tabelle 33: Spezielle Beschreibung zu Feld im LOOKC-Block (Teil 1 von 2)

Bedeutung	Inhalt	Länge	Distanz	Typ
Verbindung Schlüsselfeld-Set – Verweis auf den ersten Schlüssel, in dem das Feld Schlüsselfeld ist – Setverweis (kurz) auf den zu diesem Schlüssel gehörenden Set		2 1	14 16	binär binär
Zusatzanzeige – Die Ziffernanzahl ist gerade (nur relevant für Feldtyp = dezimal gepackt)	(Bit) 80	1	17	binär

Tabelle 33: Spezielle Beschreibung zu Feld im LOOKC-Block (Teil 2 von 2)

1 Hier sind Kombinationen möglich

Spezielle Beschreibung zu Schlüssel im LOOKC-Block

Bedeutung	Inhalt	Länge	Distanz	Typ
Satzverweis (lang)		2	0	binär
Filler		2	2	binär
Schlüssellänge		1	4	binär
Schlüsseldetails[1] – DUPLICATES NOT ALLOWED – DUPLICATES ALLOWED – DESCENDING – ASCENDING – Indextabelle für Schlüssel – impliziter Set – expliziter Set	(Bit) 80 00 40 00 20 10 00	1	5	binär
Anzahl der Felder, die den Schlüssel bilden		1	6	binär
Satzverweis (kurz)		1	7	binär
Position des 1. Schlüsselfeldes im Satz		2	8	binär
Feldlänge		2	10	binär
Feldtyp (vgl. Feldbeschreibung)		1	12	binär
Reserve		5	13	binär

Tabelle 34: Spezielle Beschreibung zu Schlüssel im LOOKC-Block

1 Hier sind Kombinationen möglich

8.5.2 LOOKC-Parameterbeschreibung

Alle LOOKC-Funktionen benutzen die folgenden CALL-DML-Parameter:

Funktionsname: **LOOKC**

sowie die CALL-DML-Parameter

- Funktionswahl
- Zusatzwahl
- Satzbereich
- Spezialparameter-1
- Spezialparameter-2

Die einzelnen Parameter sind in der Übersicht (LOOKC-Tabellen) ab Seite 278 ausführlich beschrieben.

Funktionswahl bei LOOKC-Aufrufen

Die Funktionswahl ist immer 6 Zeichen lang und besteht aus einer Kombination folgender Symbole:

ALL	alle Elemente sollen bearbeitet werden
FRT	from-to: Folge von Elementen
FST	first: erstes Element
ITM	Feld
KEY	Schlüssel
LIS	Liste von Elementen
MEM	Membersätze
NAM	Elementname wird gegeben oder gesucht
NXT	nächstes Element
OM-	Owner-/Membersatzart
OWN	Ownersatzart
REC	Satzart
RLM	Realm
SET	Set
SPC	specified: Element wird angegeben

Zusatzwahl bei LOOKC-Aufrufen

- Formatgebundene Zusatzwahl:

  Die formatgebundene Darstellung der Zusatzwahl ist bei der LOOKC-Funktion immer 6 Zeichen lang und besteht aus einer Kombination der Symbole

```
AGG         Vektor oder Datengruppe
FST         first
ITM         Feld
KEY         Schlüssel
MEM         Member
NXA         Next part of answer (nächster Antwortteil)
OWN         Owner
REC         Satzart
SET         Set
SPC         specified (angegeben)
␣␣␣         Leerzeichen
```

- Formatfreie Zusatzwahl

  Auch bei den LOOKC-Funktionen gibt es eine formatfreie Darstellung der Zusatzwahl. Die Regeln finden Sie auf Seite 203, die Symbole entsprechen der formatgebundenen Zusatzwahl bei LOOKC.

  Beispiele

  () oder (␣␣␣␣) entspricht ␣␣␣␣␣␣␣

  (SET,ITM) entspricht ITMSET

Die zugelassenen Kombinationen von Funktionswahl und Zusatzwahl und ihre Bedeutungen finden Sie in der Übersicht auf Seite 278.

Spezialparameter-2 (SPP2) bei LOOKC-Aufrufen

Die LOOKC-Funktionswahl ALLRLM zeigt alle Realms an, in denen Sätze einer bestimmten Satzart gespeichert sein können. Die Satzartnummer (REC-REF) der gewünschten Satzart übergeben Sie im Spezialparameter-2 (SPP2). Für LOOKC-Aufrufe hat der SPP2 folgendes Format:

Byte 1	**Byte 2**	**Byte 3...4**
Satzartnummer (REC-REF) ≤ 255	Filler	Satzartnummer (REC-REF) ≤ 2^{15}-1

Tabelle 35: Übergabeformat des Spezialparameter-2

Die Satzartnummer tragen Sie wie folgt in SPP2 ein:

– Eine Satzartnummer > 255 geben Sie immer in den Bytes 3...4 von SPP2 an. In diesem Fall müssen Sie in Byte 1 von SPP2 den für Satzartnummern ungültigen Wert X‘00‘ eintragen.

– Eine Satzartnummer ≤ 255 geben Sie wahlweise in Byte 1 oder in den Bytes 3...4 von SPP2 an.
  Wenn Sie eine Satzartnummer ≤ 255 in Byte 1 eintragen, ist der Inhalt der Bytes 3...4 nicht relevant.
  Wenn Sie eine Satzartnummer ≤ 255 in den Bytes 3...4 eintragen, müssen Sie in Byte 1 den für Satzartnummern ungültigen Wert X‘00‘ eintragen.

8.5.3 LOOKC-Tabellen

Folgende Abkürzungen werden in der Übersicht verwendet:

I	Input
O	Output
I,O	Output verschieden von Input
AR	Realm-Reference (binär; Länge: 1 byte)
RR	Recordreference (binär; die Länge der kurzen Referenz beträgt 1 byte, die Länge der langen Referenz beträgt 2 byte.
SR	Setreference (binär; die Länge der kurzen Referenz beträgt 1 byte, die Länge der langen Referenz beträgt 2 byte)
IR	Position des Feldes im Satz (binär; Länge: 2 byte)
K	Keyreference (binär; Länge: 2 byte)

Im Folgenden sind die einzelnen LOOKC-Tabellen jeweils auf zwei gegenüberliegenden Seiten dargestellt.

Bedeutung der Funktion	Funktions-wahl	Zusatz-wahl	Satzbereich (LOOKC-Block)		
			generelle Beschreibung (38 Bytes)		
			externer Name	Reference 1	Reference 2
	Zeichen	formatfrei	Zeichen	binär	binär
	6		30	2	2

LOOKC nach einem Namen

angegebener Name (des angegebenen Typs)	SPCNAM	()	I	I,O [1) 3)]	O [2)]
erster Name	FSTNAM	()	O	O [1)]	O [2)]
nächster Name	NXTNAM	()	I,O	O [1)]	O [2)]
alle Namen, beginnend mit dem angegebenen	ALLNAM	(SPC)	I,O	I,O [1) 3)]	O [2)]
alle Namen, beginnend mit dem ersten	ALLNAM	(FST)	O	O [1)]	O [2)]
alle Namen, nächster Antwortteil	ALLNAM	(NXA)	O	O [1)]	O [2)]
Owner-/Member-Satznamen eines Sets	OM-NAM	(SET)	O	I(SR)O(RR)	O (Null)
FROM-TO-Namen	FRTNAM	()	I,O	I,O [1) 3)]	O [2)]
FROM-TO-Namen, nächster Antwortteil	FRTNAM	(NXA)	O	O [1)]	O [2)]
Liste der angegebenen Namen	LISNAM	()	I	I,O [1) 3)]	O [2)]

LOOKC nach einem Realm

Liste aller Realms mit dem angegebenen Satztyp	ALLRLM	(REC)	O	O (AR)	
Liste aller Realms mit dem angegebenen Satzart, nächster Antwortteil	ALLRLM	(RECNXA)	O	O (AR)	

```
1) AR bei Datentyp=Realm      2) NULL bei Datentyp=Realm      3) Input nur bei Datentyp
   RR bei Datentyp=Satzart       NULL bei Datentyp=Satzart       = Feld
   SR bei Datentyp=Set           NULL bei Datentyp=Set
   RR bei Datentyp=Feld          IR   bei Datentyp=Feld
   RR bei Datentyp=Schlüssel     IR   bei Datentyp=Schlüssel
```

					Spezial-Parameter-1	Spezial-Parameter-2
				Spezielle Beschreibung (18 Bytes)		
Datentyp	gleiche Namen	Ergebnis	Reserve		Zahl der LOOKC-Blöcke	Satz-reference
binär	binär	binär	binär		binär	binär
1	1	1	1		2	4

I,O		O		Nur bei Datentyp=Feld oder Schlüssel; spezielle Feldbeschreibung wird übergeben.		
I,O		O				
I,O		O				
I,O		O			I	
I,O		O			I	
O		O			I	
					I (=2)	
I,O		O			I	
O		O			I	
I,O		O			I	

O		O		O	I	I (RR)
O		O		O	I	

Bedeutung der Funktion	Funktions-wahl	Zusatzwahl	Satzbereich (LOOKC-Block)		
			generelle Beschreibung (38 Bytes)		
			externer Name	Reference 1	Reference 2
	Zeichen	formatfrei	Zeichen	binär	binär
	6		30	2	2

LOOKC nach einer Satzart

angegebene Satzarten	SPCREC	( )	O	I (RR)	
erste Satzart	FSTREC	( )	O	O (RR)	
nächste Satzart	NXTREC	( )	O	I,O (RR)	
Ownersatzart des angegebenen Set	OWNREC	(SET)	O	O (RR)	I (SR)
Membersatzart des angegebenen Set	MEMREC	(SET)	O	O (RR)	I (SR)
Liste der angegebenen Satzarten	LISREC	( )	O	I (RR)	

LOOKC nach einem Set

angegebener Set	SPCSET	( )	O	I (SR)	
erster Set	FSTSET	( )	O	O (SR)	
nächster Set	NXTSET	( )	O	I,O (SR)	
erster Set der Ownersatzart	FSTSET	(OWNREC)	O	O (SR)	I (RR)
nächster Set der Ownersatzart	NXTSET	(OWNREC)	O	I,O (SR)	I (RR)
erster Set der Membersatzart	FSTSET	(MEMREC)	O	O (SR)	I (RR)
nächster Set der Membersatzart	NXTSET	(MEMREC)	O	I,O (SR)	I (RR)
alle Sets, in denen die angegebene Satztart Owner ist	ALLOWN	(REC)	O	O (SR)	I (RR)
alle Sets, in denen die angegebene Satzart Owner ist, nächster Antwortteil	ALLOWN	(RECNXA)	O	O (SR)	
alle Sets, in denen die angegebene Satzart Member ist	ALLMEM	(REC)	O	O (SR)	I (RR)
alle Sets, in denen die angegebene Satztart Member ist, nächster Antwortteil	ALLMEM	(RECNXA)	O	O (SR)	
Liste der angegebenen Sets	LISSET	( )	O	I (SR)	

					Spezial-Parameter-1	Spezial-Parameter-2
				Spezielle Beschreibung (18 Bytes)		
Datentyp	gleiche Namen	Ergebnis	Reserve		Zahl der LOOKC-Blöcke	Satz-reference
binär	binär	binär	binär		binär	binär
1	1	1	1		2	4

O		O		O		
O		O		O		
O		O		O		
O		O		O		
O		O		O		
O		O		O	I	

		O		O		
		O		O		
		O		O		
		O		O		
		O		O		
		O		O		
		O		O		
		O		O	I	
		O		O	I	
		O		O	I	
		O		O	I	
		O		O	I	

Bedeutung der Funktion	Funktions-wahl	Zusatz-wahl	Satzbereich (LOOKC-Block)		
			generelle Beschreibung (38 Bytes)		
			externer Name	Reference 1	Reference 2
	Zeichen	formatfrei	Zeichen	binär	binär
	6		30	2	2

LOOKC nach einem Feld

angegebenes Feld der angegebenen Satzart	SPCITM	(REC)	O	I (RR)	I (IR)
erstes Feld der angegebenen Satzart	FSTITM	(REC)	O	I (RR)	O (IR)
nächstes Feld der angegebenen Satzart	NXTITM	(REC)	O	I (RR)	I,O (IR)
alle Felder der angegebenen Satzart	ALLITM	(REC)	O	I (RR)	O (IR)
alle Felder der angegebenen Satztart im nächsten Antwortteil	ALLITM	(RECNXA)	O	O (RR)	O (IR)
alle Felder der angegebenen Zusammenstellung	ALLITM	(AGG)	O	I (RR)	I,O (IR)
alle Felder der angegebenen Zusammenstellung, nächster Antwortteil	ALLITM	(AGGNXA)	O	O (RR)	O (IR)
FROM-TO-Felder	FRTITM	()	O	I (RR)	I,O (IR)
FROM-TO-Felder, nächster Antwortteil	FRTITM	(NXA)	O	O (RR)	O (IR)
Liste der angegebenen Felder	LISITM	()	O	I (RR)	I (IR)

							Spezial-Parameter-1	Spezial-Parameter-2
				Spezielle Beschreibung (18 Bytes)				
Datentyp	gleiche Namen	Ergebnis	Reserve	Filler	Stufen-nummer	Restliche Beschreibung	Zahl der LOOKC-Bl.	Satz-reference
binär	binär	binär	binär	binär	binär		binär	binär
1	1	1	1	4	1	13	2	4

O		O	O	O	I,O[1]	O		
O		O	O	O	O	O		
O		O	O	O	I,O	O		
O		O	O	O	O	O	I	
O		O	O	O	O	O	I	
O		O	O	O	I,O	O	I	
O		O	O	O	O	O	I	
O		O	O	O	I,O	O	I	
O		O	O	O	O	O	I	
O		O	O	O	I,O	O	I	

1 Falls die angegebene Stufennummer (Input) nicht vorhanden ist, liefert UDS/SQL die nächste gefundene Stufennummer als Output.

Bedeutung der Funktion	Funktions-wahl	Zusatz-wahl	Satzbereich (LOOKC-Block)		
			generelle Beschreibung (38 Bytes)		
			externer Name	Reference 1	Reference 2
	Zeichen	formatfrei	Zeichen	binär	binär
	6		30	2	2

LOOKC nach einem Schlüssel

erster Schlüssel des angegebenen Set	FSTKEY	(SET)		I (SR)	O (K)
nächster Schlüssel des angegebenen Set	NXTKEY	(SET)		I (SR)	I,O (K)
erster Schlüssel im angegebenen Set, in dem angegebenes Feld vorhanden ist	FSTKEY	(ITMSET)		I (SR)	O (K)
nächster Schlüssel des Feldes im angegebenen Set	NXTKEY	(ITMSET)		I (SR)	I,O (K)
erster Schlüssel des Feldes	FSTKEY	(ITM)		O (SR)	O (K)
nächster Schlüssel des Feldes	NXTKEY	(ITM)		I,O (SR)	I,O (K)
alle Schlüssel des angegebenen Set	ALLKEY	(SET)		I (SR)	O (K)
alle Schlüssel des angegebenen Set; nächster Antwortteil	ALLKEY	(SETNXA)		O (SR)	O (K)
alle Schlüssel des angegebenen Feldes	ALLKEY	(ITM)		O (SR)	O (K)
alle Schlüssel des angegebenen Feldes im angegebenen Set	ALLKEY	(ITMSET)		I (SR)	O (K)
alle Schlüssel des angegebenen Feldes (im angegebenen Set); nächster Antwortteil	ALLKEY	(ITMNXA)		O (SR)	O (K)

									Spezial-Parameter-1	Spezial-Parameter-2
				Spezielle Beschreibung (18 Bytes)						
Datentyp	gleiche Namen	Ergebnis	Reserve	Satzreference (lang)	vgl. Key-Beschreibung	Satzreference (kurz)	Feldreference	vgl. Key-Beschreibung	Zahl der LOOKC-Blöcke	Satzreference
binär	binär	binär	binär	binär		binär	binär		binär	binär
1	1	1	1	2	5	1	2	8	2	4

O		O		O (RR)	O	O (RR)	O (IR)	O		
O		O		O (RR)	O	O (RR)	O (IR)	O		
O		O		I (RR)	O	I (RR)	I (IR)	O		
O		O		I (RR)	O	I (RR)	I (IR)	O		
O		O		I (RR)	O	I (RR)	I (IR)	O		
O		O		I (RR)	O	I (RR)	I (IR)	O		
O		O		O (RR)	O	O (RR)	O (IR)	O	I	
O		O		O (RR)	O	O (RR)	O (IR)	O	I	
O		O		I (RR)	O	I (RR)	I (IR)	O	I	
O		O		I (RR)	O	I (RR)	I (IR)	O	I	
O		O		O (RR)	O	O (RR)	O (IR)	O	I	

Bedeutung der Funktion	Funktions-wahl	Zusatz-wahl	Satzbereich (LOOKC-Block)		
			generelle Beschreibung (38 Bytes)		
			externer Name	Reference 1	Reference 2
	Zeichen	formatfrei	Zeichen	binär	binär
	6		30	2	2

LOOKC nach den Feldern eines Schlüssels

erstes Feld des angegebenen Schlüssels	FSTITM	(KEY)	O	O (RR), I (SR)	I (K), O (IR)
nächstes Feld des Schlüssels	NXTITM	(KEY)	O	O (RR), I (SR)	I (K), O (IR)
alle Felder des Schlüssels	ALLITM	(KEY)	O	O (RR), I (SR)	I (K), O (IR)
alle Felder des angegebenen Schlüssels; nächster Antwortteil	ALLITM	(KEYNXA)	O	O (RR)	O (IR)

										Spezial-Parameter-1	Spezial-Parameter-2
				Spezielle Beschreibung (18 Bytes) bei der							
				Eingabe					Ausgabe		
Datentyp	gleiche Namen	Ergebnis	Reserve	Satzref. (lang)	vgl. Key-Beschreibung	Satzref. (kurz)	Feldreference	vgl. Keybeschreibung	vgl. Feldbeschreibung	Zahl der LOOKC-Blöcke	Satzreference
binär	binär	binär	binär	binär		binär	binär			binär	binär
1	1	1	1	2	5	1	2	8	18	2	4

O		O							O		
O		O		I (RR)		I (RR)	I (IR)		O		
O		O							O	I	
O		O							O	I	

8.6 Beispiele

DMLTEST:

Die folgenden LOOKC-Beispiele sind mit dem Programm DMLTEST erstellt (Seite 303).

```
(IN ): * **********LOOKC NACH NAMEN ****************
(IN ): SET LOOKC,ALLNAM,FST
(IN ): DEF RECORD,DATTYP,D=34,L=1
(IN ): * *******14 LOOKC-BLOECKE************
(IN ): SET SPP1=X'0010'
(IN ): * ******** DATENTYP OHNE ANGABE **********
(IN ): M DATTYP,X'00'
(IN ): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT):   FCOD :LOOKC ..  FOPT :ALLNAM..  SOPT :FST       UINF :          RECN :
(OUT):   SETN :          RLMN :          ITMN :          RECA :          SPP1 :..
(OUT):   SPP2 :VERKAUF   SPP3 :          SUBS :ADMIN
(IN ): EX
(OUT): RECORD - AREA      :
(OUT): ABGEGEBENE-BEST               .......                    AKT-BESTAND
(IN ): SH RECORD,L=840
(OUT): RECORD - AREA      :
(OUT): ABGEGEBENE-BEST               .......                    AKT-BESTAND
(OUT):       .......     ............. STATISTIK                     ...+........
(OUT):         KENNZ-NICHT-LIEFERBAR         ............              MENGE
(OUT):               ............               FARB-NR                          ..........
(OUT): ..              FARB-BEZ                        ............              MAT-ABK
(OUT):                       ............             MAT-BEZ                          ..
(OUT): ..........              LIEFER-NR                      ............
(OUT): LIEFER-NAME                   ............              LIEFER-PLZ
(OUT):       ............              LIEFER-STADT                    ............
(OUT):         LIEFER-STRASSE                ............              LIEFER-HAUSNR
(OUT):               ............
(IN ): SH RECORD,L=840,FORM=X
(OUT): RECORD - AREA      :
(OUT): C1C2C7C5C7C5C2C5D5C560C2C5E2E3404040404040404040404040400000F0000030000404040
(OUT): 4040404040404040404040404040404040C1D2E360C2C5E2E3C1D5C4404040404040404040404040404040
(OUT): 4040404040400009004804000040404040402020100000600000080000000040E2E3C1E3C9E2E3C9
(OUT): D2404040404040404040404040404040404040404040400009004E020000000000000040404040404040
(OUT): 4040404040404040D2C5D5D5E960D5C9C3C8E360D3C9C5C6C5D9C2C1D940404040404040404040400009
(OUT): 0056020000000A0C000040404040404040404040404040404040D4C5D5C7C540404040404040404040404040
(OUT): 4040404040404040404040404040000A000002000000000000004040404040404040404040404040404040
(OUT): C6C1D9C260D5D940404040404040404040404040404040404040404040000B0000020000001120
(OUT): 0000404040404040404040404040404040C6C1D9C260C2C5E940404040404040404040404040404040404040
(OUT): 4040404040400B000202000000102000004040404040404040404040404040D4C1E360C1C2D240
(OUT): 4040404040404040404040404040404040404040404040404040000C000002000000122100004040404040
(OUT): 4040404040404040D4C1E360C2C5E940404040404040404040404040404040404040404040404040000C
(OUT): 0001020000001321000040404040404040404040404040404040D3C9C5C6C5D960D5D940404040404040
(OUT): 4040404040404040404040404040404040000D000002000000080B000040404040404040404040404040404040
```

```
(OUT): D3C9C5C6C5D960D5C1D4C5404040404040404040404040404040404040000D0005020000000080B
(OUT): 0000404040404040404040404040404040D3C9C5C6C5D960D7D3E940404040404040404040404040404040
(OUT): 4040404040404000D00230200000000000000040404040404040404040404040D3C9C5C6C5D960E2
(OUT): E3C1C4E340404040404040404040404040404040404040000D00270200000000000000404040404040
(OUT): 4040404040404040D3C9C5C6C5D960E2E3D9C1E2E2C5404040404040404040404040404040404040000D
(OUT): 0045020000000000000040404040404040404040404040404040D3C9C5C6C5D960C8C1E4E2D5D9404040
(OUT): 4040404040404040404040404040404040000D00630200000000000000404040404040404040404040404040
```

```
(IN ): S RECA=C' '
(IN ): * ********* ALLE REALMS *************
(IN ): M DATTYP,X'01'
(IN ): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT):   FCOD :LOOKC ..  FOPT :ALLNAM..  SOPT :FST       UINF :          RECN :
(OUT):   SETN :          RLMN :          ITMN :          RECA :          SPP1 :..
(OUT):   SPP2 :VERKAUF   SPP3 :          SUBS :ADMIN
(IN ): EX
(OUT): RECORD - AREA      :
(OUT): ARTIKELRLM                    .......                    AUFTRAGSRLM
(IN ): SH RECORD,L=840
(OUT): RECORD - AREA      :
(OUT): ARTIKELRLM                    .......                    AUFTRAGSRLM
(OUT):       .......                   BESTELLRLM                    .......
(OUT):         HAUSHALT                        .......                    KLEIDUNG
(OUT):                .......                    LEBENSMITTEL                .......
(OUT):                  SCHREIBWAREN                   .......                    SPIELE-H
(OUT): OBBY                    .......                    SPORT                          ..
(OUT): .....                     SUCHRLM                         .......
(OUT):
(OUT):
(OUT):
(OUT):
(IN ): SH RECORD,L=840,FORM=X
(OUT): RECORD - AREA      :
(OUT): C1D9E3C9D2C5D3D9D3D4404040404040404040404040404040404040000B0000010000404040
(OUT): 4040404040404040404040404040404040C1E4C6E3D9C1C7E2D9D3D4404040404040404040404040404040
(OUT): 4040404040400003000001000040404040404040404040404040404040404040C2C5E2E3C5D3D3D9
(OUT): D3D4404040404040404040404040404040404040404040000040000010000404040404040404040404040
(OUT): 4040404040404040C8C1E4E2C8C1D3E3404040404040404040404040404040404040404040404040400006
(OUT): 0000010000404040404040404040404040404040404040404040D2D3C5C9C4E4D5C7404040404040404040
(OUT): 4040404040404040404040404040400005000001000040404040404040404040404040404040404040404040
(OUT): D3C5C2C5D5E2D4C9E3E3C5D3404040404040404040404040404040404040000800000100004040 40
(OUT): 4040404040404040404040404040404040E2C3C8D9C5C9C2E6C1D9C5D5404040404040404040404040404040
(OUT): 4040404040400000A000001000040404040404040404040404040404040404040E2D7C9C5D3C560C8
(OUT): D6C2C2E840404040404040404040404040404040404040000900000100004040404040404040404040404040
(OUT): 4040404040404040E2D7D6D9E340404040404040404040404040404040404040404040404040400007
(OUT): 0000010000404040404040404040404040404040404040404040E2E4C3C8D9D3D4404040404040404040
(OUT): 4040404040404040404040404040400000C000001000040404040404040404040404040404040404040404040
(OUT): 4040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040
(OUT): 4040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040
```

```
(OUT): 4040404040404040404040404040404040404040404040404040404040404040404040404040
(OUT): 4040404040404040404040404040404040404040404040404040404040404040404040404040
(OUT): 4040404040404040404040404040404040404040404040404040404040404040404040404040
(OUT): 4040404040404040404040404040404040404040404040404040404040404040404040404040
(OUT): 4040404040404040404040404040404040404040404040404040404040404040404040404040
```

Die folgenden Programmierbeispiele sind erstellt mit COBOL, FORTRAN und Assembler.

COBOL:

Das COBOL-Programm geht von der CALL-DML-Variante (CALL30) aus. Es führt in einer Schleife eine FTCH7A-Anweisung mit einer Suchbedingung aus. Satzname, Feldname und Wert des Feldes werden am Bildschirm abgefragt. Die Länge des Wertes wird dabei mit 10 Zeichen vorausgesetzt. Die ersten 80 Byte des ersten Treffersatzes werden am Bildschirm ausgegeben.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CALLDML.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     TERMINAL IS T.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  FCOD            PIC X(6).
 01  FOPT            PIC X(6).
 01  SOPT            PIC X(12).
 01  UINF.
     02  SYSTEM-COMMUNICATION-LOCATIONS.
         03  DATABASE-REALM-NAME    PIC X(30).
         03  DATABASE-RECORD-NAME   PIC X(30).
         03  DATABASE-SET-NAME      PIC X(30).
         03  DATABASE-STATUS        PIC X(5).
*
     02  FILLER          PIC X.
     02  UINF-DBKEY      PIC X(4).
     02  UINF-TALLY      PIC 9(8) COMP.
     02  FILLER          PIC X(7).
     02  UINF-DBKZ       PIC X.
     02  FILLER          PIC X(8).
     02  UINF-END        PIC X(6).
*
 01  RECN            PIC X(30).
 01  SETN            PIC X(30).
 01  RLMN            PIC X(30).
 01  ITMN            PIC X(53).
 01  RECA.
     02  RECA1       PIC X(80).
     02  RECA2       PIC X(2000).
 01  SPP1            PIC X(30).
*
```

```
 01  SEARCH-EXPRESSION.
     02  SX-BEGIN    PIC X(2)     VALUE "0 ".
     02  SX-NAME     PIC X(30).
     02  SX-REL      PIC X(5)     VALUE " EQU ".
     02  SX-VALUE    PIC X(10).
     02  SX-END      PIC X(6)     VALUE " 0 END".
*
 PROCEDURE DIVISION.
*    INITIALIZE UINF.
     MOVE "UINF1*" TO UINF-END.
*

*    TRANSAKTION EROEFFNEN:
     MOVE "READYC" TO FCOD.
     MOVE "ALLRTR" TO FOPT.
     MOVE "SUBSCH" TO SPP1.
     PERFORM DMLCALL.
*
 FETCH-LOOP.
*
*    FTCH7A-ANWEISUNG:
     MOVE "FTCH7A" TO FCOD.
     MOVE "RECSEX" TO FOPT.
     MOVE SPACES TO SOPT.
     DISPLAY  "PLEASE ENTER RECORDNAME OR STOP" UPON T.
     ACCEPT   RECN    FROM T.
     IF RECN = "STOP" THEN GO TO FINISH.
*
     DISPLAY  "PLEASE ENTER ITEMNAME" UPON T.
     ACCEPT   SX-NAME FROM T.
     DISPLAY  "PLEASE ENTER ITEMVALUE" UPON T.
     ACCEPT  SX-VALUE FROM T.
     MOVE SEARCH-EXPRESSION TO ITMN.
     PERFORM DMLCALL.
     DISPLAY RECA1    UPON T.
     GO TO FETCH-LOOP.
*
 FINISH.
*    TRANSAKTION SCHLIESSEN:
     MOVE "FINISC" TO FCOD.
     MOVE "ALLRLM" TO FOPT.
     PERFORM DMLCALL.
     STOP RUN.
*
 DMLCALL.
*    CALL-DML-ANWEISUNG AUSFUEHREN:
     CALL "DML" USING FCOD FOPT SOPT UINF RECN SETN RLMN
                      ITMN RECA SPP1.
```

```
           IF DATABASE-STATUS NOT EQUAL "00000" THEN
              DISPLAY "DATABASE-STATUS: " DATABASE-STATUS
              UPON T.
      *
       DSCEXT.
      *    CALL-DML-FEHLERAUSGANG DSCEXT:
           ENTRY "DSCEXT".
           DISPLAY "DSCEXT IN COBOL WURDE ANGESPRUNGEN"
             UPON T.
           STOP RUN.
      *
```

Die folgenden Programmbeispiele gehen von der Variante (CALL8) aus. Sie sind von der Funktion her identisch; man kann damit im Dialog alle 12 CALL-DML-Parameter versorgen und somit den größten Teil aller möglichen DML-Funktionen aufrufen.
Die Eingabe ist formatfrei und darf bis zu 80 Zeichen lang sein. Jedem Parameter entspricht ein zweistelliges Kürzel, z.B. FC für FCOD, dem eine in Hochkommata eingeschlossene Zeichenfolge folgt.

Beispiel

```
FC'FTCH4'FO'RECFST'RN'SATZ1'
```

Die Parameter FCOD, FOPT und RECN werden mit der entsprechenden Zeichenfolge versorgt und gegebenenfalls mit Leerzeichen aufgefüllt.

Die Eingabe von EX oder E␣ bewirkt den Aufruf einer CALL-DML-Anweisung.

Nach Eingabe von DP (Display Parameters) werden die ersten 8 Bytes jedes Parameters am Bildschirm gezeigt; mit DR (Display Record) werden die ersten 40 Bytes des Satzbereichs gezeigt. Alle Programme werden durch die Eingabe von ST oder STOP beendet.

Die Kürzel für die Parameter bei FORTRAN und Assembler sind:

```
1.  FC (FCOD)              6.  SN  (SETN)
2.  FO  (FOPT)             7.  RL  (RLMN)
3.  SO  (SOPT)             8.  IT  (ITMN)
4.  entfällt               9.  RA  (RECA)
5.  RN  (RECN)             10. S1  (SPP1)
```

Außerdem gibt es noch folgendes Kürzel:

```
SU     Subschema
```

Der Entry DSCEXT ist in diesen Programmbeispielen nicht enthalten; ein entsprechendes Modul muss in die Programme eingebunden werden (siehe Abschnitt „Fehlerbehandlungsroutine DSCEXT der CALL-DML“ auf Seite 124).

FORTRAN:

```
      PROGRAM FORDML
      CHARACTER * 80  LINE
      CHARACTER * 80  STRING
      CHARACTER *  2  CMD
      INTEGER   *  2  POS,POSS
C
      CHARACTER *  6  FCOD /'READYC'/
      CHARACTER *  6  FOPT /'ALLUPD'/
      CHARACTER * 12  SOPT /'      '/
      CHARACTER *  8  RECN
      CHARACTER * 80  SETN,RLMN
      CHARACTER *800  ITMN,RECA
      CHARACTER * 30  SPP1 /'SUBTI1'/
C
      CHARACTER *126  UINF
      CHARACTER * 30  REALMNAME,RECNAME,SETNAME
      CHARACTER *  5  DBSTATUS      /'00000'/,
     F                FIL1       *1,
     F                DBKEY      *4,
     F                TALLY      *4,
     F                RUNID      *2,
     F                RUNREF     *2,
     F                FIL2       *3,
     F                DBREF      *1,
     F                SSITABADR  *4,
     F                FIL3       *4,
     F                USINF      *6 /'USINF*'/
      INTEGER   *  4  ZAEHLER
      COMMON   /UINF/ REALMNAME,RECNAME,SETNAME,DBSTATUS,
     F                FIL1,DBKEY,TALLY,RUNID,RUNREF,FIL2,
     F                DBREF,SSITABADR,FIL3,USINF
      EQUIVALENCE     (UINF,REALMNAME),
     F                (TALLY,ZAEHLER)

20    FORMAT(A80)
22    FORMAT(' ',A80)

C     ---------- READ INPUT ------------------------------
30    READ  (1,20) LINE
      POS = 1
C
C     ---------- GET NEW COMMAND -------------------------
80    DO 100, POS = POS,80,1
      IF (LINE(POS:POS).NE.' ') GOTO 104
100   CONTINUE
      IF (POS.GE.80) GO TO 30
```

```
104     CONTINUE
        CMD = LINE(POS:POS+1)
        DO 108, POS = POS+2,80,1
        IF (LINE(POS:POS).NE.' ') GOTO 112
108     CONTINUE
112     IF (LINE(POS:POS).NE.'''') GOTO 160
C       ---------- GET NEW STRING -------------------------
        DO 116, POSS = 1,80,1
        STRING(POSS:POSS) = ' '
116     CONTINUE
        POSS = 0
        DO 120, POS=POS+1,80,1
        IF (LINE(POS:POS).EQ.'''') GOTO 124
        POSS = POSS+1
        STRING(POSS:POSS) = LINE(POS:POS)
120     CONTINUE
124     POS = POS+1
C
160     CONTINUE

C       ---------- CHECK AND EXECUTE COMMAND -------------
        IF(CMD.EQ.'FC') GOTO 200
        IF(CMD.EQ.'FO') GOTO 204
        IF(CMD.EQ.'SO') GOTO 208
        IF(CMD.EQ.'RN') GOTO 212
        IF(CMD.EQ.'SN') GOTO 216
        IF(CMD.EQ.'RL') GOTO 220
        IF(CMD.EQ.'IT') GOTO 224
        IF(CMD.EQ.'RA') GOTO 228
        IF(CMD.EQ.'S1') GOTO 232
        IF(CMD.EQ.'SU') GOTO 232
        IF(CMD.EQ.'E ') GOTO 260
        IF(CMD.EQ.'EX') GOTO 260
        IF(CMD.EQ.'DP') GOTO 280
        IF(CMD.EQ.'DR') GOTO 288
        IF(CMD.EQ.'ST') GOTO 304
        IF(CMD.EQ.'FF') GOTO 312
C
180     FORMAT(' ANWEISUNG UNBEKANNT: ',A2)
        WRITE(2,180) CMD
        GOTO 300
C
200     FCOD = STRING( 1: 6)
        GOTO 300
204     FOPT = STRING( 1: 6)
        GOTO 300
208     SOPT = STRING( 1:12)
        GOTO 300
```

```
212      RECN = STRING( 1: 8)
         GOTO 300
216      SETN = STRING( 1:80)
         GOTO 300
220      RLMN = STRING( 1:80)
         GOTO 300
224      ITMN = STRING( 1:80)
         GOTO 300
228      RECA = STRING( 1:80)
         GOTO 300
232      SPP1 = STRING( 1:30)
         GOTO 300
260      CONTINUE
C        ---------- CALL A DML-STATEMENT --------------------
         CALL  DML (FCOD,FOPT,SOPT,UINF,RECN,SETN,
     F              RLMN,ITMN,RECA,SPP1)

264      FORMAT(' DBSTATUS: ',A5)
         IF (DBSTATUS.NE.'00000') WRITE(2,264) DBSTATUS
         GOTO 300
278      FORMAT ('    FCOD:',A6,'     FOPT:',A6,'     SOPT:',A8,
     F           '   RECN:',A8,'   SETN:',A8,'   RLMN:',A8,
     F           '   ITMN:',A8,'   RECA:',A8,'   SPP1:',A8 )
280      WRITE(2,278) FCOD,FOPT,SOPT(1:8),RECN,SETN(1:8),RLMN(1:8),
     F                ITMN(1:8),RECA(1:8),SPP1(1:8)
         GOTO 300
288      WRITE(2,22) RECA
C
300      IF (POS.GE.80) GO TO 30
         GOTO 80
304      CONTINUE
         STOP
C
C        ---------- PRODUCE A P-ERROR ----------------------
312      POS = POS/(POS-POS)
         GOTO 30
         END
```

Assembler:

Wenn Sie mit Assembler arbeiten, bauen Sie die Parameterleiste der CALL-DML mit A-Konstanten selbst auf. Sie können dazu die Makros DSCAL, DSCAP, DSCDF und DSCPA verwenden (siehe Abschnitt „CALL-DML-Assembler-Makros“ auf Seite 259).

```
ASSDML   START
         BALR  12,0
         USING *,12
         PRINT NOGEN
         SPACE
         MVC   UINF+120(6),='USINF*'
         SPACE
READ     RDATA LINEL,ERROR,84
         XR    4,4
         LH    4,LINEL
         SH    4,=H'4'
         LA    3,LINE
         LA    4,0(3,4)        R4 AT END OF INPUT
         MVI   0(4),' '
NEXT     DS    0H
BLANKS   DS    0H              IGNORE BLANKS
         CR    3,4
         BNL   READ
         CLI   0(3),' '
         BNE   MOVCMD
         LA    3,1(3)
         B     BLANKS
         SPACE
MOVCMD   MVC   CMD,0(3)        GET COMMAND
         LA    3,1(3)
BLANK2   LA    3,1(3)          IGNORE BLANKS
         CR    3,4
         BH    EXECMD
         CLI   0(3),' '
         BE    BLANK2
         CLI   0(3),''''
         BNE   EXECMD
MSTRING  DS    0H              READ A STRING
         MVC   STRING,STRING-1
         LA    5,STRING
         LA    3,1(3)
MSTR1    CR    3,4
         BNL   EXECMD
         CLI   0(3),''''
         BE    MSTREND
         MVC   0(1,5),0(3)
```

```
         LA    3,1(3)
         LA    5,1(5)
         B     MSTR1
MSTREND  LA    3,1(3)
         SPACE
EXECMD   DS    0H              CHECK AND EXECUTE COMMAND
         CLC   CMD,='FC'
         BE    PAR1
         CLC   CMD,='FO'
         BE    PAR2
         CLC   CMD,='SO'
         BE    PAR3
         CLC   CMD,='RN'
         BE    PAR5
         CLC   CMD,='SN'
         BE    PAR6
         CLC   CMD,='RL'
         BE    PAR7
         CLC   CMD,='IT'
         BE    PAR8
         CLC   CMD,='RA'
         BE    PAR9
         CLC   CMD,='S1'
         BE    PAR10
         CLC   CMD,='SU'
         BE    PAR11
         CLC   CMD,='E '
         BE    EXDML
         CLC   CMD,='EX'
         BE    EXDML
         CLC   CMD,='DP'
         BE    DISPAR
         CLC   CMD,='DR'
         BE    DISREC
         CLC   CMD,='ST'
         BE    STOP
         WROUT UNKNOWN,ERROR   MESSAGE IF COMMAND IS WRONG
         B     NEXT
         SPACE
PAR1     MVC   FCOD,STRING
         B     NEXT
PAR2     MVC   FOPT,STRING
         B     NEXT
PAR3     MVC   SOPT,STRING
         B     NEXT
PAR5     MVC   RECN,STRING
         B     NEXT
PAR6     MVC   SETN,STRING
```

```
         B     NEXT
PAR7     MVC   RLMN,STRING
         B     NEXT
PAR8     MVC   ITMN,STRING
         B     NEXT
PAR9     MVC   RECA(80),STRING
         B     NEXT
PAR10    MVC   SPP1,STRING
         B     NEXT
EXDML    DS    0H              CALL A DML-STATEMENT
         DSCAP PARAM=PARADR
         SPACE
         CLC   DBSTATUS(5),='00000'
         BE    NEXT
         WROUT MESSTAL,ERROR
         SPACE
         B     NEXT
DISPAR   DS    0H
         MVC   POUT1,FCOD
         MVC   POUT2,FOPT
         MVC   POUT3,SOPT
         MVC   POUT5,RECN
         MVC   POUT6,SETN
         MVC   POUT7,RLMN
         MVC   POUT8,ITMN
         MVC   POUT9,RECA
         MVC   POUT10,SPP1
         WROUT PAROUT,ERROR
         B     NEXT
DISREC   WROUT RECAOUT,ERROR
         B     NEXT
         SPACE
STOP     DS    0H
ERROR    DS    0H
         TERM
         SPACE
MESSTAL  DC    X'0018404040'
         DC    C'DBSTATUS: '
MESSTA   DS    CL5
         DC    C'     '
         DS    0H
LINEL    DC    C'    '
LINE     DS    CL80
UNKNOWN  DC    X'0050404040'
         DC    C'ANWEISUNG UNBEKANNT : '
CMD      DS    CL2
         DC    C' '
```

```
STRING   DS    CL80
         SPACE
PARADR   DSCDF FCOD,FOPT,SOPT,UINF,RECN,SETN,RLMN,ITMN,RECA,SPP1,      C
               SUFFIX=ADR     DEFINITION OF OPERAND LIST
         SPACE
FCOD     DC    CL6'READYC'
FOPT     DC    CL6'ALLUPD'
SOPT     DC    CL20'             '
RECN     DS    CL8
SETN     DS    CL80
RLMN     DS    CL80
ITMN     DS    CL160
RECAOUT  DC    X'002D404040'   WROUT FIRST 40 BYTES OF RECA
RECA     DS    CL800
SPP1     DS    CL30
         SPACE
UINF     DS    0F              UDS USER INFORMATION AREA
         DSCPA USERINF
DBSTATUS EQU   USREDMCO        DEFINED IN MACRO DSCPA
         SPACE
PAROUT   DC    Y(PAROUTEN-PAROUT)
         DC    X'404040'
         DC    C'   FCOD:'
POUT1    DC    C'       '
         DC    C'     FOPT:'
POUT2    DC    C'       '
         DC    C'     SOPT:'
POUT3    DS    CL8
         DC    C'   RECN:'
POUT5    DS    CL8
         DC    C'   SETN:'
POUT6    DS    CL8
         DC    C'   RLMN:'
POUT7    DS    CL8
         DC    C'   ITMN:'
POUT8    DS    CL8
         DC    C'   RECA:'
POUT9    DS    CL8
         DC    C'   SPP1:'
POUT10   DS    CL8
PAROUTEN EQU   *
         LTORG
         DS    0F
         DC    C'**** END OF ASSDML  ****'
         END
```

9 Testen von DML-Funktionen mit DMLTEST

Mit dem Programm DMLTEST können Sie

– einzelne DML-Funktionen im Dialog testen
– Testprozeduren ablaufen lassen
– auf jede beliebige Datenbankkonfiguration zugreifen
– mit KDBS zusammenarbeiten

Die folgende Beschreibung ist aufgeteilt in

– Einführung
– Kommandos des Programmes DMLTEST
– DML-Anweisungen in DMLTEST
– Ablauf von DMLTEST
– Fehlermeldungen

9.1 Einführung

Das Dialog-Programm DMLTEST bietet folgende Möglichkeiten:

- umgebungsspezifisches Verhalten
- eine weitgehend formatfreie Eingabe
- bei den CALL-Schnittstellen die Möglichkeit, jedes einzelne Byte der CALL-Parameterleiste explizit anzusprechen
- vollkommene Unabhängigkeit von den benutzten Datenbanken
- vollkommene Unabhängigkeit von UDS/SQL-Varianten. Alle UDS/SQL-spezifischen Teile werden nachgeladen. Es ist kein Binden erforderlich.
- Belegung von Feldern mit symbolischen Namen
- eine Reihe von komfortablen Kommandos zur Ausgabe von Ergebnissen und zur Steuerung des Ablaufs
- Bearbeitung vordefinierter Kommandofolgen (temporär definiert oder in System-Dateien hinterlegt)
- beliebig viele Benutzeranschlüsse
- SYSTEM-Kommando (Programmunterbrechung mit Wechsel in den System-Modus)
- TRACE-Kommando (TRACE von DMLTEST-Kommandos)
- DMLTEST unterstützt sowohl die CALL8- als auch die CALL30-Schnittstelle der CALL-DML
- EDT-Anschluss
- DMLTEST ist XS-fähig

Standardfunktionen

Das Programm arbeitet, wenn Sie nichts anderes angegeben haben, mit linked-in DBH und COBOL-DML mit CALL8-Schnittstelle.

Intern benützt DMLTEST die Schnittstelle der CALL-DML; der DML-Funktionsumfang entspricht dem Funktionsumfang der CALL-DML (siehe Abschnitt „Besonderheiten der CALL-DML“ auf Seite 92).

Die Zuweisung von SYSDTA

Wenn SYSDTA auf die Datensichtstation zugewiesen ist, geht DMLTEST davon aus, dass Sie im Dialog arbeiten wollen. Sie können dann auch direkt auf Unterbrechungen reagieren.

Wenn SYSDTA auf eine Datei zugewiesen ist, erwartet DMLTEST, dass Sie nicht im Dialog arbeiten wollen, sondern einen automatischen Betrieb wünschen (Prozedur oder Stapelauftrag). Auf Unterbrechungen haben Sie keinen Einfluss. Der automatische Ablauf erfolgt solange bis eins der folgenden Ereignisse erkannt wird:

– HALT-Kommando

– EOF

– ein Fehler tritt auf

– SYSDTA wird auf die Datensichtstation gelegt

Im Fehlerfall gibt DMLTEST eine Meldung aus. Sofern bereits ein Aufruf an den UDS/SQL-DBH erfolgt ist, erzeugt DMLTEST sprachspezifisch die Anweisung FINISH WITH CANCEL. Anschließend beendet sich DMLTEST.

Kommandofolge zum Starten von DMLTEST

Das Programm DMLTEST ist eine CALL-DML-Anwendung, die entweder mit dem linked-in DBH (Default) oder mit dem independent DBH zusammenarbeitet (siehe DMLTEST-Kommando DBH auf Seite 317). DMLTEST wird mit dem Kommando START-UDS-DMLTEST (ALiasname: DMLTEST) gestartet. Welche Kommandos vor dem eigentlichen DMLTEST-Start benötigt werden, hängt von der DBH-Variante ab. Sie ist dann jeweils analog zur Kommandofolge, die vor dem Aufruf von CALL-DML-Anwenderprogrammen benötigt werden.

1. linked-in DBH

```
 /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=konfigurationsname
 /ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=SSITAB-nachladebibliothek
[/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=dateiname] ——————————  (1)
 /CREATE-FILE FILE-NAME=konfname.DBSTAT,SUPPRESS-ERR=*FILE-EXISTING
 /CREATE-FILE FILE-NAME=konfname.DBSTAT.SAVE,SUPPRESS-ERR=*FILE-EXISTING
 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
 /START-UDS-DMLTEST
```

   Die Kommandofolge ist analog zum Start von linked-in CALL-DML-Anwenderprogrammen (siehe Handbuch „Datenbankbetrieb“, Abschnitt DBH-Startkommandos).

   (1) Mit dem Linknamen PPFILE wird eine Datei zugewiesen, aus der der DBH die Ladeparameter einlesen soll. Die Ladeparameter können auch in einem PLAM-Bibliothekselement oder in einer Prozedurdatei stehen. Die hierfür benötigten Zuweisungskommandos sind dem Handbuch „Datenbankbetrieb“, Abschnitt DBH-Startkommandos zu entnehmen.

2. independent DBH

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=konfigurationsname
/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=SSITAB-nachladebibliothek
/START-UDS-DMLTEST
```

Die Kommandofolge ist analog zum Start von CALL-DML-Anwenderprogrammen, die den independent DBH nutzen (siehe Abschnitt „COBOL-Programm starten“ auf Seite 103ff).

9.2 Kommandos des Programms DMLTEST

Übersicht über die DMLTEST-Kommandos und allgemeine Regeln

Kommando	Funktion
ADD *name*,*value*[,*condition*]	addieren
CONTINUE	nach einer Unterbrechung Verarbeitung fortsetzen
DBH {INDEPENDENT | INLINKED}	DBH-Variante auswählen Standardwert: INLINKED
{DECLARE | DCL} *name*[,*length*]	ein Feld im Arbeitsbereich festlegen
DEFINE {RCODE | RECORD | *parameter*},*name*[,*distance*][,*length*]	ein Feld in dem angegebenen Bereich definieren
DELETE *name*[,*condition*]	eine Prozedur oder ein Feld löschen
DISPLAY {RECA | RECORD | RCODE | TIME}[,*distance*][,*length*][,*form*] [,*condition*]	den angegebenen Bereich oder Wert nach jeder DML-Anweisung auf SYSOUT ausgeben
{DISPOFF | DOFF}[{RECA | RECORD | RCODE | TIME}][,*condition*]	DISPLAY-Funktion ausschalten
DO *name*[,*repetition*][,*condition*]	eine Prozedur starten
EDT	den Dateiaufbereiter EDT aufrufen
END	Prozedurdefinition beenden
ESCAPE[*condition*]	eine Unterbrechung beenden oder eine Prozedur abbrechen
EXECUTE[*repetition*][,*condition*]	Ausführen einer DML-Anweisung

Tabelle 36: Übersicht über die DMLTEST-Kommandos (Teil 1 von 3)

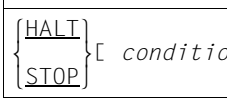
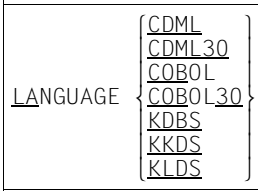
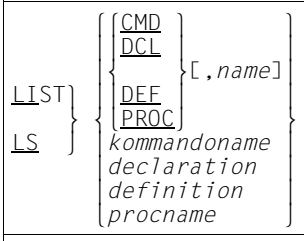
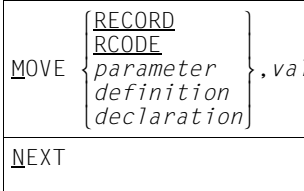
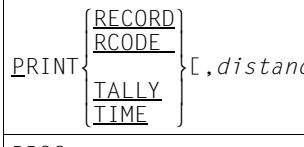
Kommando	Funktion
	das Programm DMLTEST beenden
HELP[*condition*]	Informationen zum letzten Kommando anfordern oder die letzte DML-Anweisung anfordern
	die Datenbehandlungssprache auswählen Standardwert: COBOL
LEAVE[*condition*]	einen Prozeduraufruf abbrechen
	die angegebene Information auf SYSOUT ausgeben
	die angegebenen Bereiche mit Werten versorgen
NEXT	auf Unterbrechungen reagieren oder das aktuelle Kommando abbrechen
PERFORM *name*[,*filename*][,*condition*]	ein Modul aufrufen
	den angegebenen Bereich oder Wert nach jeder DML-Anweisung auf SYSLST ausgegeben
PROC *procname*	eine Prozedurdefinition eröffnen
	PRINT-Funktion ausschalten

Tabelle 36: Übersicht über die DMLTEST-Kommandos (Teil 2 von 3)

Kommando	**Funktion**
PROT[{ON | OFF | OUT}][,*condition*]	Protokollfunktion ein- oder ausschalten Standardwert: PROT ON
{REMARK | *} *literal*	Kommentarzeilen einfügen
RUN *filename*[,*repetition*][,*condition*]	eine Kommando- oder Anweisungsfolge starten, die in einer ISAM-Datei abgelegt ist.
SET[*parameter*[({*n* | X'*n*'})]=]*value*,...	die CALL-DML-Parameterleiste mit Werten versorgen
SHOW { RCODE | TALLY | {RECORD | PARAM[,*name*]} | {DCL | DEF | PROC},*name* | *parameter* | *declaration* | *definition* | *procname* } [,*distance*] [,*length*] [,*form*] [,*condition*]	den angegebenen Bereich in der angegebenen Form auf SYSOUT ausgeben
SUBSCHEMA IS *subschema*	Subschema auswählen
SYSTEM[*condition*]	in Systemmodus gehen
ITRACE[{ON | OFF}][,*repetition*][,*condition*]	Kommandos und Anweisungen während der Verarbeitung protokollieren auf dem Bildschirm Standardwert: TRACE ON
WAIT[*condition*]	eine Unterbrechung bewirken

Tabelle 36: Übersicht über die DMLTEST-Kommandos (Teil 3 von 3)

Die Kommandos für DMLTEST können Sie

- im Dialog eingeben,
- in temporären DMLTEST-Prozeduren hinterlegen,
- in BS2000-Prozedurdateien hinterlegen,
- in BS2000-Enterdateien hinterlegen,
- in BS2000-ISAM-Systemdateien hinterlegen.

Sie können beliebig viele Kommandos auf einmal eingeben. Sie dürfen nur die folgende Eingabelänge nicht überschreiten:

1 Bildschirminhalt und 256 byte im Unterbrechungsfall

Beispiel

```
      /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=VERSAND
      /ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=LMS.SSITAB
      /ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=UDSDBB.PP.FILE
1)    /CREATE-FILE FILE-NAME=VERSAND.DBSTAT,SUPPRESS-ERRORS=*FILE-EXISTING
      /CREATE-FILE FILE-NAME=VERSAND.DBSTAT.SAVE,SUPPRESS-ERRORS=*FILE-EXISTING
      /ASSIGN-SYSDTA TO-FILE=*SYSCMD
      /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.8A00
      /START-UDS-DMLTEST

2)    LANG COB
3)    DISPLAY RCODE,COND=RCODE NE C'00000'
4)    PROT ON
5)    DISPLAY RECA,L=80
6)    SUBSCHEMA IS ADMIN
7)    READY
8)    E
      %  UDSO215 UDS STARTET UDS/SQL V2.8 (LINKED-IN), DATE=2015-06-28
      (ILL2038,11:26:35/OYA2)
       %  UDSO746 UDS-PUBSET-DEKLARATION (CURRENT) FOLGT
      (ILL1746,11:26:35/OYA2)
       OYA2: UDS-PUBSET-JV:  :SQL2:$XXXXXXXX.PUBSDECL.ALL
       OYA2: PUBSETS:        *
       OYA2: DEFAULT PUBSET: SQL2
       OYA2: ----------------------------------------
       %  UDSO722 UDS AUFTRAG ADD RLOG 150628092634 WIRD AUSGEFUEHRT
      (ILL1283,11:26:35/OYA2)
       %  UDSO354 UDS ALOG-KONSISTENZPUNKT FUER VERSAND
      (ILL1307,11:26:35/OYA2)
       OYA2: ALOG-CKPT OMITTED: DB WITHOUT ALOG-LOGGING.
       OYA2: MAXDB          =      1
       OYA2: TRANSACTION    =      1
       OYA2: SUBSCHEMA      =      1
       OYA2: 2KB-BUFFER-SIZE=      1
       OYA2: 4KB-BUFFER-SIZE=      1
       OYA2: 8KB-BUFFER-SIZE=      0
       OYA2: LOG            = PUBLIC
       OYA2: LOG-2          = NO
       OYA2: LOG-SIZE       = (    192,    192)
       OYA2: RESERVE        = NONE
       OYA2: WARMSTART      = STD
       OYA2: CONSOLE        = NO
       OYA2: STDCKPT        = YES
       OYA2: LOCK           = STD
       OYA2: PRIVACY-CHECK  = OFF
       OYA2: CONFNAME       = $XXXXXXXX.VERSAND
       OYA2: DATABASES OF CONFIGURATION:
```

```
       OYA2:   $XXXXXXXX.VERSAND    ,EXCLUSIVE-UPD     ,*SYSTEM
       %  UDS0356 UDS DURCHFUEHRUNG DER AUFTRAEGE FUER VERSAND  TERMINATED
      (ILL1309,11:26:35/OYA2)
       RECORD - AREA      :
      ......................................................................

9)    SYS
      %  IDA0199 PROGRAMM-UNTERBRECHUNG AUF ADRESSE X'00010E04', AMODE=31
      /MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=*TIME(TIMEOUT=*STD)
10)   /RESUME-PROGRAM
11)   EDT
12)   @RET
      EDT ENDED. DMLTEST CONTINUES
13)   RUN UDSANW.P.DMLTEST1
      .
      .
      .
 DB-PARAMS (FIRST 8B)
 FCOD :ACCPTC.. FOPT :DB-KEY..  SOPT :          UINF :          RECN :ARTIKEL
 SETN :         RLMN :          ITMN :          RECA :........  SPP1 :ADMIN
 SPP2 :....     SPP3 :          SUBS :ADMIN
 .
 .
 .
 DB-PARAMS (FIRST 8B)
 FCOD :ACCPTC.. FOPT :RLMDBK..  SOPT :          UINF :KLEIDUNG  RECN :ARTIKEL
 DMLTEST: SCREEN OVERFLOW.   STATUS IS BREAK.

14)   ESC
15)   FINISH
      E

16)   HALT
      DMLTEST NORMAL TERMINATION
      %  UDS0354 UDS ALOG-KONSISTENZPUNKT FUER VERSAND(ILL1307,11:26:31/OYA2)
      OYA2: ALOG-CKPT 20150628092639: FIXED ( ALOG-NR:000000005, START-CKPT:
      20150628092638 ).
      %  UDS0758 ANZAHL DER DML-ANWEISUNGEN UND I/O-ZAEHLER PRO DATENBANK
      (ILL1758,17:05:20/OYA2)
       OYA2: DATABASE NAME           DMLS   LOG READ  PHYS READ  LOG WRITE
      PHYS WRITE
       OYA2: ----------------------------------------------------------------
      --------
       OYA2: VERSAND                 75       325        99        91
      63
      %  UDS0213 UDS NORMAL BEENDET MIT **************27 DML-STATEMENTS
      2015-06-28 (ILLY033,11:26:31/OYA2)
```

1) Siehe „Kommandofolge zum Starten von DMLTEST“ (Seite 305)
2) Anwendungssprache festlegen
3) Festlegen, in welchem Fall der DATABASE-STATUS ausgegeben werden soll
4) Protokollierung einschalten
5) Festlegen, dass RECA nach jeder DML-Anweisung ausgegeben werden soll
6) Subschema zuweisen
7) READY-Anweisung aufbauen
8) READY-Anweisung ausführen
9) Steuerung an Betriebssystem übergeben
10) Steuerung an DMLTEST übergeben
11) EDT als Unterprogramm aufrufen
12) Steuerung an DMLTEST übergeben
13) DMLTEST-Prozedur aufrufen
14) DMLTEST-Prozedur abbrechen
15) Transaktion beenden
16) Programm DMLTEST beenden

Allgemeine Regeln

variable müssen Sie bei der Benutzung eines Formats durch den aktuellen Wert ersetzen. Dabei sind folgende Kategorien zu unterscheiden:

Variable	**aktueller Wert**
definition *declaration* *kommandoname* *procname*	Diese Namen müssen in den entsprechenden DMLTEST-Kommandos definiert werden.
literal	setzt sich aus beliebigen Zeichen zusammen. Handelt es sich nicht um eine alphanumerische Darstellung, müssen Sie es mit Hochkommata begrenzen. Falls ein Literal mit alphanumerischer Darstellung Leerzeichen enthält, müssen Sie es ebenfalls mit Hochkommata begrenzen.

Tabelle 37: Variablen von DMLTEST

In den Formaten der DMLTEST-Kommandos werden außerdem folgende Schlüsselwortparameter verwendet:

Schlüsselwortparameter	**Standardwert**
value: [VAL=]{ *literal* | [*n1*]{C | X | P}[L*n2*]'*literal*' | 'literal' }	-
distance: [DIST=]{ *n* | X'*n*' }	D=0
length: [LNG=]{ *n* | X'*n*' }	L=8
name: [NAME=]*literal* maximale Länge : 20 erlaubte Zeichen: 1. Stelle A-Z, ab 2. Stelle A-Z,-,0-9	-
filename: [{OML= | FILE=}]*literal*	OML= DMLTEST.MODLIB bei PERFORM
repetition: [{REP= | STEP=}]{ *n* | X'*n*' }	4 bei TRACE 1 in allen anderen Fällen
form: [FORM=] {C | X | D}	F=C
condition: [{CASE= | COND=}]{ RECORD{(*n*) | (X'*n*')} | TIME | RCODE | *definition* }{EQ | NE | LT | GT | LE | GE}*value*	-

Tabelle 38: Schlüsselwortparameter mit Standardwerten

definition muss in Hochkommata gesetzt werden

C alphanumerische Darstellung
X sedezimale Darstellung
D Dump-Format
P gepackte Darstellung
n Ganzzahl
n1 Multiplikationsfaktor
n2 Länge des Literals

Die Variable *parameter* in den DMLTEST-Kommandos kann durch folgende Werte ersetzt werden (siehe auch Tabelle 23 auf Seite 201):

Stellung in der CALL-Parameterleiste	Schlüsselwort für den Parameter	
	bei CDML	bei KDBS
1	FCOD	OP
2	FOPT	RE
3	SOPT	DB
4	UINF	AR
5	RECN	FS
6	SETN	SI
7	RLMN	KB
8	ITMN	KE
9	RECA	RT
10	SPP1	ST
11	SPP2	FSI[2]
12	SPP3	–
	SEX[1]	–
	SUBS	–

Tabelle 39: Werte bei CDML und KDBS

1 Redefinition von ITMN: Diesen Parameter müssen Sie verwenden, wenn Sie die Aufbereitung eines Suchausdrucks von DMLTEST-Syntax auf CALL-DML-Syntax wünschen (siehe Kommando SET auf Seite 335).

2 Redefinition von SI: Diesen Parameter müssen Sie verwenden, wenn Sie die Aufbereitung eines Suchausdrucks von DMLTEST-Syntax auf KDBS-Syntax wünschen (siehe Kommando SET auf Seite 335).

Sie können beliebig viele Kommandos hintereinander schreiben. Das Trennzeichen ist „;“.

Vordefinierte Felder

Die nachfolgend beschriebenen Felder sind vordefiniert und können mit den DMLTEST-Kommandos LIST, SHOW, MOVE und ADD angesprochen werden.

Folgende Felder sind immer definiert:

RCODE	enthält Anweisungscode und Statuscode
TALLY	enthält den Zähler für TALLYING
RECORD	enthält den Satzbereich (RECORD AREA)
TIME	enthält die Ausführungszeit der Anweisung (in Millisekunden)

Folgende Felder sind für die Verwendung der COBOL-DML definiert:

AREA-ID	enthält den Realm-Namen bei ACCEPT-2. Das Feld AREA-ID ist eine Redefinition des Bereichs für den Realm-Namen im Abschnitt „System Communication Locations“ des Benutzerinformationsbereichs (siehe Tabelle 24 auf Seite 205).
DB-KEY	enthält den DATABASE-KEY-Wert bei ACCEPT-1, ACCEPT-2, FIND/FETCH-1. Das Feld DB-KEY ist eine Redefinition des Feldes DATABASE-KEY des Benutzerinformationsbereichs (siehe Tabelle 24 auf Seite 205).
DB-KEY-LONG	enthält den DATABASE-KEY-LONG-Wert bei ACCEPT-1, ACCEPT-2, FIND/FETCH-1. Das Feld DB-KEY-LONG ist eine Redefinition des Feldes DATABASE-KEY-LONG des Benutzerinformationsbereichs (siehe Tabelle 24 auf Seite 205).
IMP-AREA-ID	enthält den Realm-Namen bei FIND/FETCH-2 und STORE...IMP bzw. STORE...IMP-LONG. Das Feld IMP-AREA-ID ist eine Redefinition des Bereichs Byte 5...34 von Spezialparameter-2 (SPP2, siehe Tabelle 26 auf Seite 210).
IMP-DB-KEY	enthält den DATABASE-KEY-Wert bei STORE...IMP. Das Feld DB-KEY ist eine Redefinition des Bereichs Byte 1...4 von Spezialparameter-2 (SPP2, siehe Tabelle 26 auf Seite 210).
IMP-DB-KEY-LONG	enthält den DATABASE-KEY-LONG-Wert bei STORE...IMP-LONG. Das Feld DB-KEY-LONG ist eine Redefinition des Bereichs Byte 35...42 von Spezialparameter-2 (SPP2, siehe Tabelle 26 auf Seite 210).

Folgende Felder sind für die Verwendung von KDBS definiert:

SC	enthält den Schutzcode
VA	enthält die Verarbeitungsart

Die Felder SC und VA sind Redefinitionen des Verständigungsbereichs RE.

ADD

Das ADD-Kommando addiert in den angegebenen Bereich.

```
ADD name,value[,condition]
```

name müssen Sie mit dem Kommando DEFINE oder DECLARE beschrieben haben.

bei DECLARE:
Der Typ ist durch ein vorhergehendes MOVE-Kommando bestimmt.

bei DEFINE:
Der Typ ist durch die Beschreibung des Bereichs bestimmt, der redefiniert wurde.

name enthält nach der Ausführung das Ergebnis der Addition.

Die Art der Addition bestimmen Sie durch den Typ von *value*, der mit dem von *name* übereinstimmen muss:

- P: gepackt
- C: entpackt
- X: sedezimal

name darf maximal folgende Länge haben:

- bei entpackter Darstellung: 16
- bei gepackter Darstellung: 10
- bei sedezimaler Darstellung: 4

value muss vom gleichen Typ sein wie der Inhalt von *name*.

Beispiele

```
***(IN ): REMARK ***ADD***
***(IN ): DEF RECORD,FELD2,L=4
***(IN ): SH FELD2,F=X
   (OUT): FELD2              :00000000
***(IN ): ADD FELD2,X'F5'
***(IN ): SH FELD2,F=X
   (OUT): FELD2              :000000F5

***(IN ): DEC FELD3,L=8
***(IN ): MOVE FELD3,8
***(IN ): SH FELD3
   (OUT): FELD3              :8
***(IN ): ADD FELD3,9
***(IN ): SH FELD3
   (OUT): FELD3              :17
```

CONTINUE

Das CONTINUE-Kommando setzt nach einer Unterbrechung die Verarbeitung fort. Lag keine Unterbrechung vor, bleibt das Kommando wirkungslos.

```
CONTINUE
```

DBH

Das DBH-Kommando wählt die DBH-Variante.

```
DBH { INDEPENDENT }
    { INLINKED    }
```

Das Kommando können Sie nur vor dem ersten SET-Kommando oder der ersten DML-Anweisung geben. Standardwert ist INLINKED.

DECLARE

Das DECLARE-Kommando legt ein Feld im Arbeitsbereich fest.

```
DECLARE }
        } name[,length]
DCL     }
```

name muss eindeutig sein und darf pro Programmlauf nur einmal vorkommen.

Der Feldtyp ist unqualifiziert und wird erst durch ein MOVE-Kommando festgelegt. Jedes weitere MOVE-Kommando kann den Typ wieder ändern.

Beispiel

```
***(IN ): REMARK ***DECLARE***
***(IN ): DEC FELD3,L=8
***(IN ): MOVE FELD3,8
***(IN ): SH FELD3
   (OUT): FELD3             :8
```

DEFINE

Das DEFINE-Kommando definiert ein Feld in dem angegebenen Bereich.

```
DEFINE {RCODE    },name[,distance][,length]
       {RECORD   }
       {parameter}
```

name muss eindeutig sein und darf pro Programmlauf nur einmal vorkommen.

Mit DEFINE können Sie Bereiche redefinieren, die Sie oft ansprechen wollen und ersparen sich eventuelle Distanz- und Längenangaben.

Mehrfache Redefinition eines Bereichs ist möglich.

Beispiel

```
***(IN ): REMARK ***DEFINE***
***(IN ): DEFINE RECORD,VORNAME,L=10
***(IN ): DEFINE RECORD,NACHNAME,D=10,L=15
***(IN ): DEFINE RECORD,GEBURTSTAG,D=25,L=10
***(IN ): DEFINE RECORD,NAMEN,L=25
***(IN ): MOVE VORNAME,ANTON
***(IN ): MOVE NACHNAME,MAIER
***(IN ): MOVE GEBURTSTAG,'12.12.50'
***(IN ): SH RECORD,L=35
   (OUT): RECORD - AREA      :ANTON.....MAIER.........12.12.50..
***(IN ): MOVE NAMEN,ALBERT
***(IN ): SH RECORD,L=35
   (OUT): RECORD - AREA      :ALBERT....MAIER.........12.12.50..
***(IN ): SH VORNAME
   (OUT): VORNAME            :ALBERT..
***(IN ): SH NACHNAME
   (OUT): NACHNAME           :MAIER...
***(IN ): SH GEBURTSTAG
   (OUT): GEBURTSTAG         :12.12.50
***(IN ): SH NAMEN,L=25
   (OUT): NAMEN              :ALBERT....MAIER.........
```

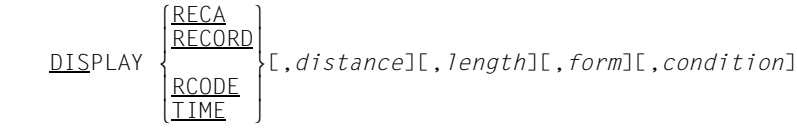
DELETE

Das DELETE-Kommando löscht eine Prozedur, Definition oder Declaration.

```
DELETE name[,condition]
```

DISPLAY

Das DISPLAY-Kommando gibt den angegebenen Bereich oder Wert nach jeder DML-Anweisung auf dem Bildschirm aus, falls die angegebene Bedingung erfüllt ist.



Es gibt folgende Ausgabeformate

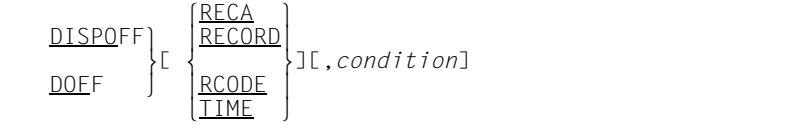
```
FORM=C   alphanumerische Zeichen (Standard)
FORM=X   sedezimal
FORM=D   sowohl C als auch X (Dump-Format)
```

Bei RCODE und TIME werden Angaben zu *distance*, *length* und *form* ignoriert.

Der Inhalt von nicht alphanumerischen Feldern kann im Allgemeinen nur im sedezimalen Format erkannt werden. Dies gilt auch für den Inhalt von nationalen Feldern (Unicode: UTF-16, PICTURE N, USAGE NATIONAL).

DISPOFF

Das DISPOFF-Kommando schaltet die angegebene oder alle vorherigen DISPLAY-Funktionen aus, falls die eventuell angegebene Bedingung erfüllt ist.



DO

Das DO-Kommando startet eine DMLTEST-Prozedur.

```
DO name[,repetition][,condition]
```

name muss in demselben Programmlauf definiert worden sein.

repetition
bestimmt, wie oft die Prozedur durchlaufen werden soll (Wiederholungsfaktor).

condition
gibt an, unter welcher Bedingung die Prozedur durchlaufen werden soll. Die Bedingung wird nur einmal vor dem ersten Durchlaufen der Prozedur geprüft, auch wenn ein Wiederholungsfaktor angegeben wurde.

Das DO-Kommando kann auch im Unterbrechungszustand gegeben werden, wird aber erst dann ausgeführt, wenn die Unterbrechung beendet wurde.

In einem DMLTEST-Lauf können gleichzeitig maximal 20 Prozeduren bearbeitet werden. Die Schachtelungstiefe geht bis zu 20 Prozeduren.

Sie können DO-WHILE-Schleifen programmieren.

```
01) PROC proc1;
02) LEAVE condition;
    .
03) .
    .
04) END
```

01) Eröffnen der Definition von *proc1*

02) Als erstes Kommando wird in *proc1* ein bedingtes LEAVE-Kommando hinterlegt.

03) Es folgen weitere Kommandos und Anweisungen der Prozedur.

04) Abschluss der Definition von *proc1*.

Mit dem Kommando

```
DO proc1,W=50000
```

wird die Bearbeitung von *proc1* angestoßen.

Die Prozedur wird solange bearbeitet, bis eins der beiden folgenden Ereignisse eintritt:

– *condition* aus dem hinterlegten LEAVE-Kommando ist erfüllt.

– Die Prozedur wurde bereits 50 000-mal abgearbeitet.

Beispiel

```
DO STORE-A,R=20,COND=RCODE EQ C'00000';
```

Zunächst wird geprüft, ob der DB-Status gleich C’00000’ ist. Ist dies der Fall, so wird die Prozedur STORE-A 20-mal durchlaufen (vorausgesetzt, sie ist definiert).

Ablauf:

I = 20 (Wiederholungsfaktor)	
Ist der DB-Status gleich C‘00000‘ ? Y	N
Solange I > 0	
STORE-A (Prozedur) bearbeiten	
I = I - 1	
Nächstes Kommando bearbeiten	

EDT

Das EDT-Kommando ruft den Dateiaufbereiter EDT als Unterprogramm auf.

Der Aufruf erfolgt im L-Modus, der nicht Unicode-fähig ist.

```
EDT
```

Mit @RETURN kehren Sie in den DMLTEST-Dialog zurück.

END

Das END-Kommando beendet eine Prozedurdefinition.

```
END
```

ESCAPE

Das ESCAPE-Kommando beendet die Bearbeitung aller begonnenen Prozeduren, Kommandofolgen und Dialogeingaben und beendet eine Unterbrechung. Das nächste Kommando wird von SYSDTA erwartet.

```
ESCAPE[ condition]
```

condition

gibt an, unter welcher Bedingung die Bearbeitung der Prozedur(en) und Kommandofolge(n) abgebrochen werden soll.

Beispiel

```
***(IN ): REMARK ***ESCAPE***
***(IN ): RUN DMLTEST.BEISPIELE1
            .
            .
            .
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :CONNEC..  FOPT :TO-SET..  SOPT :          UINF :KLEIDUNG  RECN :ARTIKEL
   (OUT): DMLTEST: SCREEN OVERFLOW.   STATUS IS BREAK.
***(IN ): ESCAPE
***(IN ): HELP
   (OUT): LAST INPUT         :ESCAPE
   (OUT): DURING BREAK-STATUS STATUS IS NORMAL
```

EXECUTE

Das EXECUTE-Kommando führt eine DML-Anweisung aus. Sie können die gleiche DML-Anweisung beliebig oft hintereinander wiederholen.

EXECUTE[*repetition*][,*condition*]

---

repetition
bestimmt, wie oft das Kommando wiederholt werden soll (Wiederholungsfaktor).

condition
gibt an, unter welcher Bedingung die DML-Anweisung ausgeführt werden soll.

Ablauf:

<table>
<tr><td colspan="3">I = Wiederholungsfaktor</td></tr>
<tr><td colspan="3">Solange I > 0</td></tr>
<tr><td></td><td colspan="2">Ist condition erfüllt?

Y N</td></tr>
<tr><td></td><td>DB-Aufruf absetzen</td><td rowspan="2">I = 0</td></tr>
<tr><td></td><td>I = I - 1</td></tr>
<tr><td colspan="3">Nächstes Kommando bearbeiten</td></tr>
</table>

Geben Sie ein EXECUTE-Kommando während einer Unterbrechung, werden *repetition* und *condition* ignoriert.

HALT

Das HALT-Kommando beendet den DMLTEST-Lauf. Eine nicht beendete Transaktion wird zurückgesetzt (Rollback).

```
{HALT}
{    }[ condition]
{STOP}
```

condition
gibt an, unter welcher Bedingung der DMLTEST-Lauf beendet werden soll.

HELP

Das HELP-Kommando gibt folgende Informationen auf dem Bildschirm aus:

- das letzte Kommando vor HELP
- den Namen des Eingabemediums, von dem das Kommando stammt
- den Status (NORMAL oder BREAK)
- eventuell eine Fehlerbeschreibung.

```
HELP[ condition]
```

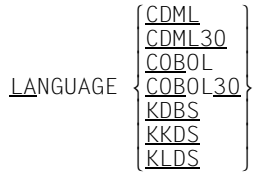
condition
gibt an, unter welcher Bedingung Informationen ausgegeben werden sollen.

Beispiel

```
***(IN ): REMARK ***HELP***
***(IN ): DEF RECORD,NACHNAME,D=10,L=15
   (OUT): DMLTEST: USED NAME. STATUS IS NORMAL.
***(IN ): HELP
   (OUT): LAST INPUT         :DEF RECORD,NACHNAME,D=10,L=15
   (OUT): FROM SYSDTA         STATUS IS NORMAL
   (OUT): ERROR DESCRIPTION  :NAME IST SCHON FUER EINE DEFINITION VERGEBEN!
```

LANGUAGE

Das LANGUAGE-Kommando wählt die Datenbehandlungssprache aus.



CDML CALL-DML; es wird die CALL8-Schnittstelle verwendet.

CDML30 CALL-DML; es wird die CALL30-Schnittstelle verwendet.

KDBS KDBS (kompatible Datenbankschnittstelle)

KKDS KDBS (kompatible Schnittstelle zur Bearbeitung von komplexen Datenstrukturen)

KLDS KDBS (kompatible Schnittstelle zur Bearbeitung von linearen Datenstrukturen)

COBOL COBOL-DML; es wird intern die CALL8-Schnittstelle verwendet.

COBOL30 COBOL-DML; es wird intern die CALL30-Schnittstelle verwendet.

Standardwert ist COBOL.

Mit dem LANGUAGE-Kommando veranlassen Sie, dass die Parameterbereiche angelegt und entsprechend initialisiert werden. Das LANGUAGE-Kommando können Sie nur vor dem ersten SET-Kommando oder der ersten DML-Anweisung geben. Deshalb ist ein Wechsel der Schnittstellen CALL8 (CDML, COBOL) und CALL30 (CDML30, COBOL30) innerhalb desselben DMLTEST-Laufs nicht möglich.

Bei LANGUAGE COBOL bzw. LANGUAGE COBOL30 können Sie die DML-Anweisungen im COBOL-DML-Format mit gewissen Einschränkungen eingeben. Diese COBOL-DML-Anweisungen werden von DMLTEST in CALL-DML-Format umgesetzt. Sie können daher auch nur den Funktionsumfang der CALL-DML benutzen. Bei LANGUAGE COBOL30 werden Namen voller Länge (30 byte) übergeben.
Das SET-Kommando können Sie auch bei LANGUAGE COBOL bzw. LANGUAGE COBOL30 benutzen.

Beispiel

```
***LANGUAGE***
***(IN ): LAN CDML
***(IN ): DISP RCODE,COND=RCODE NE C'0000'
***(IN ): DISP RECA,L=80
***(IN ): SET FCOD=C'READYC',FOPT=C'ALLEUP',SPP1=C'ADMIN                          '
          ...
```

LEAVE

Das LEAVE-Kommando beendet die Bearbeitung einer Prozedur oder Kommandofolge und gibt die Steuerung an die nächsthöhere Prozedur, Kommandofolge bzw. den Bildschirm zurück. Eine Unterbrechung wird nicht beendet.

LEAVE[*condition*]

---

condition

gibt an, unter welcher Bedingung eine Prozedur oder Kommandofolge abgebrochen werden soll.

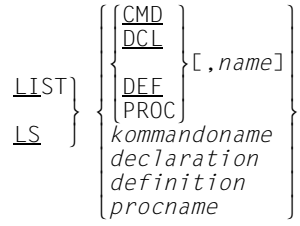
Brechen Sie mit LEAVE die Bearbeitung eines DO- oder RUN-Kommandos ab, so wird die Prozedur oder Kommandofolge unabhängig vom Wiederholungsfaktor als vollständig bearbeitet betrachtet.

Beispiel

```
***(IN ): REMARK ***LEAVE***
***(IN ): PROC TEST
***(IN ): SH RECORD,L=40
***(IN ): LEAVE
***(IN ): SH RECORD,L=40
***(IN ): END
***(IN ): DO TEST
***(IN ): SH RECORD,L=40
   (OUT): RECORD - AREA      :ANTON.....MAIER.........12.12.50.......
***(IN ): LEAVE
***(IN ): HELP
   (OUT): LAST INPUT         :LEAVE
   (OUT): FROM :   TEST       STATUS IS NORMAL
***(IN ): CONTINUE
***(IN ): HELP
***(OUT): LAST INPUT         :CONTINUE
***(OUT): FROM SYSDTA         STATUS IS NORMAL
```

LIST

Das LIST-Kommando gibt die angegebenen Informationen auf dem Bildschirm aus.



CMD, DEF, DCL, PROC
können Sie mit dem gewünschten Namen *name* kombinieren.

CMD Geben Sie keinen Namen an, werden alle Kommandonamen mit den Abkürzungen ausgegeben.

DEF Geben Sie keinen Namen an, werden alle Definitionsnamen ausgegeben.

DCL Geben Sie keinen Namen an, werden alle Deklarationsnamen ausgegeben.

PROC Geben Sie keinen Namen an, werden alle Prozedurnamen ausgegeben.

Beispiele

```
***(IN ): REMARK ***LIST***
***(IN ): LIST DCL
   (OUT): LIST OF DECLARATIONS
   (OUT): NAME      :    FELD1
   (OUT): NAME      :    FELD3
***(IN ): LIST DEF
   (OUT): LIST OF DEFINITIONS
   (OUT): NAME      :    USERGROUP
   (OUT): NAME      :    USERNAME
   (OUT): NAME      :    PASSWORD
   (OUT): NAME      :    DB-KEY
   (OUT): NAME      :    AREA-ID
   (OUT): NAME      :    IMP-DB-KEY
   (OUT): NAME      :    IMP-AREA-ID
   (OUT): NAME      :    FELD2
   (OUT): NAME      :    VORNAME
   (OUT): NAME      :    GEBURTSTAG
   (OUT): NAME      :    NACHNAME
***(IN ): LIST DEC
   (OUT): NAME      : DECLARE    (DEC  )
   (OUT): OTHER-NAME: DCL        (DC   )
   (OUT): 1. OPERAND: <NAME>
   (OUT): 2. OPERAND: <LENGTH>
```

MOVE

Das MOVE-Kommando belegt den Satzbereich, die CALL-Parameterleiste oder Felder mit Werten.

```
        ⎧RECORD     ⎫
        ⎪RCODE      ⎪
MOVE    ⎨parameter  ⎬,value[,distance][,condition]
        ⎪definition ⎪
        ⎩declaration⎭
```

parameter
gibt den gewünschten CALL-Parameter an

declaration
Die Angabe von *distance* ist nicht erlaubt.

Der Typ, den *declaration* annimmt, bestimmt sich nach dem Typ, den *value* besitzt und ist unabhängig von dem Typ, den *declaration* vorher hatte.

Der Typ von *value* bestimmt die Art der Übertragung:

Typ	Übertragung
alphanumerisch	linksbündig, mit Leerzeichen aufgefüllt, rechts abgeschnitten
numerisch gepackt	rechtsbündig, links abgeschnitten, gefüllt mit X‘ 00‘
numerisch entpackt	rechtsbündig, links abgeschnitten, gefüllt mit 0
sedezimal	linksbündig, rechts abgeschnitten

definition
wird linksbündig übertragen, eventuell ab der angegebenen Distanz. Wenn *value* kürzer als *definition* ist, wird nicht aufgefüllt.

value Es gibt folgende Typen:

```
C'-123'  alphanumerisch
ABC1     alphanumerisch
P'23'    numerisch gepackt
-12.3    numerisch entpackt
X'12'    sedezimal
```

Das MOVE-Kommando unterstützt keine nationalen Literale (Unicode: UTF-16).

RECORD, RCODE, *parameter*
wird wie *definition* übertragen.

Beispiele

```
***(IN ): REMARK ***MOVE***
***(IN ): MOVE FELD3,8
***(IN ): SH FELD3
   (OUT): FELD3              :8
***(IN ): MOVE NACHNAME,MAIER
***(IN ): SH NACHNAME
   (OUT): NACHNAME           :MAIER...
***(IN ): MOVE GEBURTSTAG,'12.12.50'
***(IN ): SH GEBURTSTAG
   (OUT): GEBURTSTAG         :12.12.50
```

NEXT

Das NEXT-Kommando bricht die Bearbeitung eines Kommandos ab und startet das nächste. Eine Unterbrechung wird beendet.

```
NEXT
```

Bei einer Unterbrechung wegen Bildschirmüberlauf wird die Ausgabe nicht mehr ausgeführt. Unterbrechen Sie ein Kommando mit Wiederholungsfaktor, wird es als vollständig bearbeitet betrachtet.

Beispiel

```
***(IN ): REMARK ***NEXT***
***(IN ): PROC TEST
***(IN ): SH RECORD,L=45
***(IN ): LIST CMD
***(IN ): SH RECORD,L=45,F=X
***(IN ): END
***(IN ): DO TEST
***(IN ): SH RECORD,L=45
   (OUT): RECORD - AREA      :ANTON.....MAIER.....12.12.50.....
***(IN ): LIST CMD
   (OUT): LIST OF COMMANDS
   (OUT): NAME      : ADD        (A    )
   (OUT): NAME      : CONTINUE   (C    )
   (OUT): NAME      : DBH        (DB   )
   (OUT): NAME      : DECLARE    (DEC  )
   (OUT): OTHER-NAME: DCL        (DC   )
   (OUT): NAME      : DEFINE     (DEF  )
   (OUT): NAME      : DELETE     (DEL  )
   (OUT): NAME      : DISPLAY    (DIS  )
   (OUT): NAME      : DISPOFF    (DISPO)
   (OUT): OTHER-NAME: DOFF       (DOF  )
   (OUT): NAME      : DO         (DO   )
   (OUT): NAME      : EDT        (ED   )
   (OUT): NAME      : END        (EN   )
   (OUT): NAME      : ESCAPE     (ES   )
   (OUT): NAME      : EXECUTE    (E    )
   (OUT): NAME      : HALT       (HALT )
   (OUT): OTHER-NAME: STOP       (STOP )
   (OUT): NAME      : HELP       (H    )
   (OUT): NAME      : LANGUAGE   (LA   )
   (OUT): DMLTEST: SCREEN OVERFLOW.   STATUS IS BREAK.
***(IN ): NEXT
***(IN ): SH RECORD,L=45,F=X
   (OUT): RECORD - AREA      :
   (OUT): C1D5E3D6D50000000000D4C1C9C5D900000000000000000000F1F24BF1F24BF1F200000000000000
   (OUT): 0000000000
```

PERFORM

Das PERFORM-Kommando ruft ein benutzereigenes Modul auf.

```
PERFORM name[,filename][,condition]
```

name muss der Name eines Moduls sein.

Das Modul muss folgende Konventionen berücksichtigen:

- R1 enthält die Adresse der CALL-Parameter-Adressleiste.
  Das 21. Wort der Adressleiste enthält binär die Zeit, die der DML-Aufruf benötigte, der zuletzt mit Zeitmessung lief (in 1/10000 Sek.).
- R13 enthält die die Adresse der Save-Area (18 Worte)
- R14 enthält die Rücksprungadresse

filename
wird *filename* nicht angegeben, so muss das Modul in der DMLTEST.MODLIB oder in Ihrer eigenen TASKLIB eingetragen sein.

condition
gibt an, unter welcher Bedingung das Modul aufgerufen werden soll.

PRINT

Das PRINT-Kommando gibt den angegebenen Bereich oder Wert nach jeder DML-Anweisung auf SYSLST aus.

```
        ⎧RECORD⎫
        ⎪RCODE ⎪
PRINT   ⎨      ⎬[,distance][,length][,form][,condition]
        ⎪TIME  ⎪
        ⎩TALLY ⎭
```

Es gibt folgende Ausgabeformate:

- FORM=C: alphanumerische Zeichen (Standard)
- FORM=X: sedezimal
- FORM=D: sowohl C als auch X (Dump-Format)

Bei RCODE und TIME werden Angaben zu *distance*, *length* und *form* ignoriert.

Der Inhalt von nicht alphanumerischen Feldern kann im Allgemeinen nur im sedezimalen Format erkannt werden. Dies gilt auch für den Inhalt von nationalen Feldern (Unicode: UTF-16, PICTURE N, USAGE NATIONAL).

PROC

Das PROC-Kommando eröffnet eine Prozedurdefinition.

```
PROC procname
```

procname
: Die Länge des Namens ist beliebig. Es werden nur die ersten 8 Zeichen als Prozeduridentifikation verwendet.

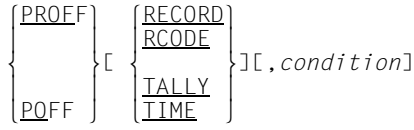
Die Eingaben, die dem PROC-Kommando folgen, werden nicht syntaktisch überprüft.

Fehlt das END-Kommando und geben Sie ein zweites PROC-Kommando ein, so wird die bisher definierte Prozedur gelöscht.

Sie dürfen in einem Programmlauf maximal 20 Prozeduren gleichzeitig bearbeiten.

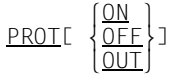
PROFF

Das PROFF-Kommando schaltet die angegebenen oder alle PRINT-Funktionen aus.



PROT

Das PROT-Kommando steuert Protokollfunktionen von DMLTEST auf SYSLST.



ON protokolliert alle Ein- und Ausgaben

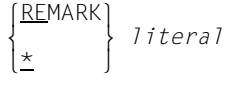
OFF schaltet die Protokollfunktion aus

OUT protokolliert nur SHOW-Ausgaben

Standardwert ist PROT ON.

REMARK

Das REMARK-Kommando fügt Kommentarzeilen ein.



literal ist beliebig. Das Zeichen ; (Kommandotrenner) dürfen Sie nur innerhalb von Hochkommata verwenden. Die Anzahl der Hochkommata muss geradzahlig sein.

RUN

Das RUN-Kommando startet eine Kommandofolge, die in einer ISAM-Datei abgelegt ist.

```
RUN filename[,repetition][,condition]
```

filename
: Die Datei *filename* muss nach dem EDTISAM-Format aufgebaut sein und folgende Eigenschaften haben:
RECFORM = V
KEYLEN = 8
KEYPOS = 5

Der Text darf maximal 256 byte lang sein und kann mehrere Kommandos enthalten.

repetition
: bestimmt, wie oft die Kommandofolge wiederholt werden soll (Wiederholungsfaktor).

condition
: gibt an, unter welcher Bedingung die Kommmandofolge ausgeführt werden soll.

Die Bearbeitung einer Kommandofolge läuft genauso ab wie eine DO-Prozedur.

In einem DMLTEST-Lauf können Sie maximal 20 verschiedene RUN-Kommandos gleichzeitig bearbeiten (Schachtelungstiefe = 20).

SET

Das SET-Kommando versorgt die Parameter der CALL-DML oder der KDBS mit Werten.

```
SET[ parameter[({ n     })]=]value,...
                { X'n'  }
```

parameter
: die einzelnen Parameter können sowohl durch ihre Stellung als auch durch ihren Namen identifiziert werden.

n
: wird als Distanz zum Anfang des angegebenen Feldes betrachtet. Geben Sie *n* an, wird nur der angegebene *value* übertragen und nicht mit Leerzeichen aufgefüllt.

  Geben Sie *n* nicht an, wird *value* linksbündig übertragen und mit Leerzeichen aufgefüllt.

Der letzte Schlüsselwortparameter bestimmt immer die Stellung der nachfolgenden Stellungsparameter. Geben Sie keinen Schlüsselwortparameter an, wird der erste angegebene Wert im FCOD eingetragen.

Bei Stellungsparametern können Sie keine Distanz angeben. *value* wird linksbündig übertragen und mit Leerzeichen aufgefüllt.

Ist *value* einschließlich *n* länger als das Feld in der Parameterleiste, so wird kommentarlos abgeschnitten.

Das SET-Kommando unterstützt keine nationalen Literale (Unicode: UTF-16).

Das SET-Kommando bearbeitet bei CDML bzw. CDML30 und KDBS unterschiedliche Parametersätze.

Suchausdrücke mit dem SET-Kommando

Sie können mit DMLTEST Suchausdrücke in der CALL-DML-Syntax (Parameter SEX) oder in der KDBS-Syntax (Parameter FSI) aufbereiten.

Es gelten folgende Regeln:

– Der Suchausdruck muss als alphanumerisches Feld in Assembler-Syntax eingegeben werden

– Syntaktische Einheiten müssen durch mindestens ein Leerzeichen voneinander getrennt sein.

– Vergleichswerte müssen in Assembler-Syntax eingegeben werden, sie werden entsprechend aufbereitet.

– Hochkommata innerhalb von Vergleichswerten sind erlaubt. Sie müssen doppelt angegeben werden.

– Bei CDML (CALL8-Schnittstelle) werden Feldnamen auf 8 byte verlängert.
Bei CDML30 (CALL30-Schnittstelle) werden Feldnamen auf 30 byte verlängert.

Verwenden Sie die CALL-DML-Syntax, müssen Sie den *parameter* ITMN angeben.

Soll DMLTEST den Suchausdruck aufbereiten, müssen Sie den *parameter* SEX verwenden.

Beispiel

```
***(IN ): SET ITMN=C'O GROESSE  GTH 40 O END'
***(IN ): SH ITMN,L=30
   (OUT): ITMN  CONTAINS     :O GROESSE  GTH 40 O END

***(IN ): SET SEX=C'GROESSE GTH CL2''40'' END'
***(IN ): SH SEX,L=30
   (OUT): SEX   CONTAINS     :O GROESSE  GTH 40 O END
```

Verwenden Sie die KDBS-Syntax, müssen Sie den *parameter* SI angeben.
Soll DMLTEST den Suchausdruck aufbereiten, müssen Sie den *parameter* FSI verwenden.

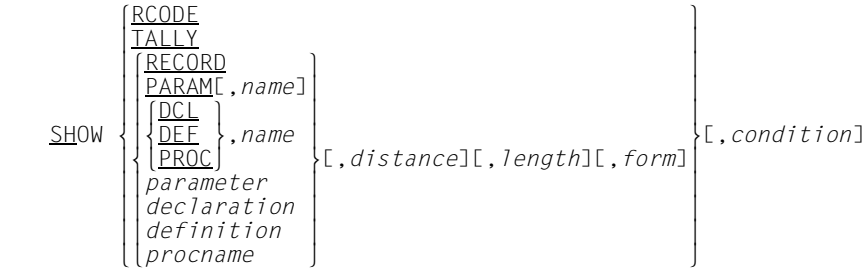
Beispiel

```
***(IN ): SET SI=C'(FELD1   EQ     WERT1)'
***(IN ): SH SI,L=40
   (OUT): SI    CONTAINS     :(FELD1   EQ     WERT1)

***(IN ): SET FSI=C'(FELD1 EQ CL10''     WERT1'')'
***(IN ): SH FSI,L=40
   (OUT): FSI   CONTAINS     :(FELD1   EQ     WERT1)
```

SHOW

Das SHOW-Kommando gibt den angegebenen Bereich auf dem Bildschirm aus.



parameter, *declaration*, *definition*, *procname*

Geben Sie einen entsprechenden Namen an, prüft DMLTEST in folgender Reihenfolge, ob der angegebene Name
- ein CALL-Parameter
- eine Definition
- ein Prozedurname
- eine Declaration

ist, und gibt den entsprechenden Inhalt am Bildschirm aus.

PARAM

Der Inhalt aller CALL-DML-Parameter wird ausgegeben.
Wenn die angegebene Länge *length* größer ist als die tatsächliche Länge, wird die angegebene Länge auf die tatsächliche Länge gekürzt. Die tatsächliche Länge ist je nach Parameter an der CALL30-Schnittstelle größer als an der CALL8-Schnittstelle.

PROC Die angegebene Prozedur wird in alphanumerischer Darstellung ausgegeben, *form* wird ignoriert.

DCL *distance* und *length* werden ignoriert.

Der Inhalt von nicht alphanumerischen Feldern kann im Allgemeinen nur im sedezimalen Format erkannt werden. Dies gilt auch für den Inhalt von nationalen Feldern (Unicode).

Beispiel

```
***(IN ): REMARK ***SHOW***
***(IN ): SH RECORD,L=40
   (OUT): RECORD - AREA      :ALBERT....MAIER.........12.12.50.......
***(IN ): SH RECORD,D=10,L=15,FORM=X
   (OUT): RECORD - AREA      :D4C1C9C5D900000000000000000000
***(IN ): SH RECORD,D=10,L=15,FORM=C
   (OUT): RECORD - AREA      :MAIER.........
***(IN ): SH RECORD,D=10,L=15,FORM=D
   (OUT): RECORD - AREA      :
   (OUT): D4C1C9C5 D9000000 00000000 000000   MAIER.........
```

SUBSCHEMA

Das SUBSCHEMA-Kommando wählt das Subschema aus, das Sie mit DMLTEST bearbeiten wollen.

SUBSCHEMA IS *subschema*

---

subschema
: Name des Subschemas, das Sie mit DMLTEST bearbeiten wollen

Weitere Informationen zu SUBSCHEMA finden Sie auf den Seiten 342 und 364.

SYSTEM

Das SYSTEM-Kommando unterbricht den Programmlauf und gibt die Steuerung an das System. Sie können Systemkommandos eingeben.

SYSTEM[*condition*]

---

condition
: gibt an, unter welcher Bedingung die Steuerung an das System gehen soll.

Mit /RESUME geben Sie die Steuerung an DMLTEST zurück.

TRACE

Das TRACE-Kommando protokolliert alle bearbeiteten Kommandos und DML-Anweisungen auf dem Bildschirm oder schaltet diese Funktion aus.

```
TRACE[ {ON } ][,repetition][,condition]
       {OFF}
```

ON, OFF
: wenn keine der Angabe vorhanden ist, ist ON Standard.

repetition
: bestimmt, wie viele Kommandos protokolliert werden sollen, bis die Bearbeitung unterbrochen wird.
repetition wird bei OFF ignoriert.

  Der Standardwert ist 4

condition
: gibt an, unter welcher Bedingung die TRACE-Funktion ausgeführt werden soll.

Beispiel

```
***(IN ): REMARK ***TRACE***
***(IN ): TRACE ON,R=6
***(IN ): RUN DMLTEST.BEISPIELE1
   (OUT): **CURRENT COMMAND**:RUN DMLTEST.BEISPIELE1
***(IN ): * ********** DMLTEST.BEISPIELE1 *****************
   (OUT): **CURRENT COMMAND**:
   (OUT): * ********** DMLTEST.BEISPIELE1 *****************
***(IN ): *
   (OUT): **CURRENT COMMAND**:
   (OUT): *
***(IN ): DISPOFF RECORD
   (OUT): **CURRENT COMMAND**:DISPOFF RECORD
***(IN ): REMARK *** BEISPIEL ACCEPT FORMAT 1 ***
   (OUT): **CURRENT COMMAND**:REMARK *** BEISPIEL ACCEPT FORMAT 1 ***
***(IN ): FIND 4 ARTIKEL
   (OUT): **CURRENT COMMAND**:FIND 4 ARTIKEL
   (OUT): DMLTEST: TRACE-LIMIT. STATUS IS BREAK.
***(IN ): C
***(IN ): EX
   (OUT): **CURRENT COMMAND**:EX
***(IN ): ACCEPT DB-KEY FROM CURRENCY
   (OUT): **CURRENT COMMAND**:ACCEPT DB-KEY FROM CURRENCY
***(IN ): EX
   (OUT): **CURRENT COMMAND**:EX
```

```
***(IN ): SH PARAM
   (OUT): **CURRENT COMMAND**:SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :ACCPTC..  FOPT :DB-KEY..  SOPT :          UINF :          RECN :ARTIKEL
   (OUT):   SETN :(MIN-BES  RLMN :          ITMN :          RECA :ANTON...  SPP1 :ADMIN
   (OUT):   SPP2 :....4     SPP3 :          SUBS :ADMIN
***(IN ): SH DB-KEY,FORM=X
   (OUT): **CURRENT COMMAND**:SH DB-KEY,FORM=X
   (OUT): DB-KEY             :09000004
***(IN ): REMARK *** BEISPIEL ACCEPT FORMAT 2 ***
   (OUT): **CURRENT COMMAND**:REMARK *** BEISPIEL ACCEPT FORMAT 2 ***
   (OUT): DMLTEST: TRACE-LIMIT. STATUS IS BREAK.
***(IN ): TRACE OFF
```

WAIT

Das WAIT-Kommando bewirkt eine Unterbrechung.

WAIT[*condition*]

---

condition
gibt an, unter welcher Bedingung eine Unterbrechung bewirkt werden soll.

Ist SYSDTA auf eine Datei zugewiesen, wird die Unterbrechung mit CONTINUE beendet. Sie haben keine Möglichkeit, einzugreifen.

9.3 Die DML-Anweisungen in DMLTEST

Sie können die DML-Anweisungen im Format der COBOL-DML eingeben (siehe Abschnitt „Anweisungen der COBOL-DML“ auf Seite 139). DMLTEST setzt diese Anweisungen in das CALL-DML-Format um. Je nachdem, ob Sie beim LANGUAGE-Kommando das Schlüsselwort COBOL oder COBOL30 angegeben haben (siehe Seite 325), versorgt DMLTEST die CALL8- oder die CALL30-Schnittstelle.

9.3.1 Übersicht über die Unterschiede zwischen den DMLTEST-DML- und den COBOL-DML-Anweisungen

Anweisung	COBOL-DML	DMLTEST-DML
ACCEPT Format 1:	*feldname-1*	DB-KEY, DB-KEY-LONG
ACCEPT Format 2:	*feldname-2* *feldname-3*	AREA-ID DB-KEY, DB-KEY-LONG
FIND/FETCH Format 1:	*feldname* OR PRIOR/NEXT	DB-KEY, DB-KEY-LONG Diese Klausel können Sie in DMLTEST in der Form SET FOPT,DBKPRI/DBKNXT verwenden, nachdem Sie die FIND1/FTCH1- bzw. FIND1L/FTCH1L-Anweisung in COBOL-DML-Syntax (ohne EXECUTE) eingegeben haben.
FIND/FETCH Format 2:		Wenn die Satzart in mehreren Realms gespeichert werden kann, müssen Sie IMP verwenden: ...*satzname*[IMP]
FIND/FETCH Format 3:		Sie können USING erweitern: ...[{IN / OF} *satzname*]
FIND/FETCH Format 4:	*feldname*	–
FIND/FETCH Format 7:	*feldname-1* OR PRIOR/NEXT	TALLY Diese Klausel können Sie in DMLTEST in der Form SET FOPT,...ITP/...ITN verwenden, nachdem Sie die FIND7A/FTCH7A-Anweisung in COBOL-DML-Syntax (ohne EXECUTE) eingegeben haben.
GET		Sie können die Anweisung erweitern: ...[{IN / OF} *satzname*]
IF Format 1: Format 2:		NEXT SENTENCE, ELSE und NOT dürfen Sie nicht verwenden
MODIFY		Die Anweisung hat folgenden Zusatz: ... {IN / OF} *satzname*

Tabelle 40: Unterschiede zwischen COBOL-DML- und DMLTEST-DML-Anweisungen

Anweisung	COBOL-DML	DMLTEST-DML
STORE		Wenn die Satzart in mehreren Realms gespeichert werden kann oder der Database-Key-Wert durch den Anwender vergeben wird (DDL-Klausel LOC MODE), müssen Sie den Zusatz IMP bzw.IMP-LONG verwenden: ...*satzname*[IMP]... bzw....*satzname*[IMP-LONG]..

Tabelle 40: Unterschiede zwischen COBOL-DML- und DMLTEST-DML-Anweisungen

Wenn Sie LANGUAGE COBOL bzw. LANGUAGE COBOL30 angegeben haben, können Sie für die Subschemazuweisung folgendes Format verwenden:

```
SUBSCHEMA[ IS] subschemaname
```

Diese Anweisung müssen Sie vor der READY-Anweisung geben.

Die SUBSCHEMA-Anweisung dient außerdem der Zuordnung von DML-Anweisungen zu Transaktionsketten bzw. zu den jeweiligen Datenbanken, wenn Sie Multi-DB-Transaktionen eröffnen.

9.3.2 Die DML-Anweisungen

ACCEPT

ACCEPT (Format 1):

Den Database-Key-Wert legt DMLTEST im vordefinierten Feld DB-KEY bzw. DB-KEY-LONG ab.

Beispiel

```
***(IN ): REMARK *** BEISPIEL ACCEPT FORMAT 1 ***
***(IN ): FIND 4 ARTIKEL
***(IN ): EX
***(IN ): ACCEPT DB-KEY FROM CURRENCY
***(IN ): EX
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :ACCPTC..  FOPT :DB-KEY..  SOPT :          UINF :          RECN :ARTIKEL
   (OUT):   SETN :          RLMN :          ITMN :          RECA :........  SPP1 :ADMIN
   (OUT):   SPP2 :....      SPP3 :          SUBS :ADMIN
***(IN ): SH DB-KEY,FORM=X
   (OUT): DB-KEY             :09000004
```

ACCEPT (Format 2):

```
ACCEPT AREA-ID FROM[ { satzname  } ] REALM-NAME
                     { setname   }
                     {           }
                     { DB-KEY    }
                     { DB-KEY-LONG }
```

In das vordefinierte Feld DB-KEY bzw. DB-KEY-LONG müssen Sie den Database-Key-Wert binär eingegeben haben. Das Ergebnis (Realm-Name) legt DMLTEST im vordefinierten Feld AREA-ID ab.

Beispiel

```
***(IN ): REMARK *** BEISPIEL ACCEPT FORMAT 2 ***
***(IN ): ACCEPT AREA-ID FROM DB-KEY REALM-NAME
***(IN ): EX
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :ACCPTC..  FOPT :RLMDBK..  SOPT :          UINF :KLEIDUNG  RECN :ARTIKEL
   (OUT):   SETN :          RLMN :          ITMN :          RECA :........  SPP1 :ADMIN
   (OUT):   SPP2 :....      SPP3 :          SUBS :ADMIN
***(IN ): SH AREA-ID
   (OUT): AREA-ID            :KLEIDUNG
```

CONNECT

```
CONNECT[ satzname] TO { setname-1,... }
                      { ALL           }

      [ RETAINING CURRENCY FOR { setname-2,... } ]
                               { SETS          }
```

Beispiel

```
***(IN ): REMARK ***BEISPIEL CONNECT ***
***(IN ): FIND 1 ARTIKEL
***(IN ): EX
***(IN ): CONNECT ARTIKEL TO MIN-BESTAND-ERREICHT
***(IN ): EX
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :CONNEC..  FOPT :TO-SET..  SOPT :          UINF :          RECN :ARTIKEL
   (OUT):   SETN :(MIN-BES  RLMN :          ITMN :          RECA :........  SPP1 :ADMIN
   (OUT):   SPP2 :....      SPP3 :          SUBS :ADMIN
***(IN ): FETCH 1 ARTIKEL WITHIN MIN-BESTAND-ERREICHT
***(IN ): EX
***(IN ): SH RECORD,LNG=48
   (OUT): RECORD - AREA      :OOOOO110SOMMERKLEID MIT JACKE
```

DISCONNECT

DISCONNECT (Format 1):

```
DISCONNECT[ satzname] FROM { setname,... }
                           { ALL         }
```

Beispiel

```
***(IN ): REMARK *** BEISPIEL DISCONNECT FORMAT 1 ***
***(IN ): FETCH 1 ARTIKEL WITHIN MIN-BESTAND-ERREICHT
***(IN ): EX
***(IN ): DISCONNECT ARTIKEL FROM MIN-BESTAND-ERREICHT
***(IN ): EX
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :DISCON..  FOPT :FRMSET..  SOPT :          UINF :          RECN :ARTIKEL
   (OUT):   SETN :(MIN-BES  RLMN :          ITMN :          RECA :00000110  SPP1 :ADMIN
   (OUT):   SPP2 :....      SPP3 :          SUBS :ADMIN
***(IN ): FINISH
***(IN ): EX
```

DISCONNECT (Format 2):

```
DISCONNECT ALL FROM setname,...
```

ERASE

```
ERASE satzname[ { PERMANENT } MEMBERS]
               { SELECTIVE }
               { ALL       }
```

Beispiel

```
***(IN ): REMARK *** BEISPIEL ERASE ***
***(IN ): FETCH LAST ARTIKEL
***(IN ): E
***(IN ): ERASE ARTIKEL ALL
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :ERASEC..  FOPT :ALLMEM..  SOPT :          UINF :          RECN :ARTIKEL
   (OUT):   SETN :(ABGEGEB  RLMN :          ITMN :(LIEFER-  RECA :901145 0  SPP1 :
   (OUT):   SPP2 :...r      SPP3 :          SUBS :ADMIN
```

FIND/FETCH

```
{FIND }
{     } satzauswahlausdruck[ RETAINING CURRENCY FOR
{FETCH}

        {MULTIPLE                          }
        {                                  }
        {[REALM][{SETS       }][RECORD]    }]
        {        {           }             }
        {        {setname,...}             }
        {MULTIPLE                          }
```

Formate des Satzauswahlausdrucks:

FIND/FETCH (Format 1):

```
[satzname] DATABASE-KEY IS {DB-KEY     }
                           {           }
                           {DB-KEY-LONG}
```

In das vordefinierte Feld DB-KEY bzw. DB-KEY-LONG müssen Sie den Database-Key-Wert binär eingegeben haben.

Beispiel

```
***(IN ): REMARK *** BEISPIEL FETCH 1 ***
***(IN ): M DB-KEY,X'09000004'
***(IN ): FETCH DATABASE-KEY IS DB-KEY
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH1 ..  FOPT :DB-KEY..  SOPT :          UINF :KLEIDUNG  RECN :
   (OUT):   SETN :(MIN-BES  RLMN :          ITMN :          RECA :00000110  SPP1 :ADMIN
   (OUT):   SPP2 :          SPP3 :          SUBS :ADMIN
***(IN ): SH RECORD,L=80
   (OUT): RECORD - AREA      :
   (OUT): 00000110SOMMERKLEID MIT JACKE                  23010742.................rn...
```

Den Zugriff auf den vorigen oder nächsten Satz zum angegebenen Database-Key (COBOL-Parameter OR PRIOR/NEXT), für den es keinen Satz in der Datenbank gibt, können Sie dadurch realisieren, dass Sie zunächst die FIND/FETCH-Anweisung eingeben und dann vor der Ausführung noch den Parameter FOPT mit DBKPRI oder DBKNXT versorgen (z.B. FETCH DATABASE-KEY IS DB-KEY; MOVE FOPT,DBKNXT;E).

FIND/FETCH (Format 2):

```
{ANY       }
{          } satzname[ IMP]
{DUPLICATE }
```

Wenn die Satzart *satzname* in mehrere Realms gespeichert werden kann, müssen Sie den Zusatz IMP angeben, und das vordefinierte Feld IMP-AREA-ID mit dem Realm-Namen des Realms versorgen, in dem der Satz gesucht werden soll, wenn die Satzart nicht Membersatzart einer verteilbaren Liste ist.

Beispiel

```
***(IN ): REMARK *** BEISPIEL FETCH 2 ***
***(IN ): FETCH 9 ARTIKELBESCHR
***(IN ): E
***(IN ): SH RECORD
   (OUT): RECORD - AREA      :000005SO
***(IN ): S SPP2=C' '
***(IN ): M IMP-AREA-ID,'KLEIDUNG'
***(IN ): FETCH ANY ARTIKELBESCHR IMP
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH2 ..  FOPT :ANYIMP..  SOPT :          UINF :KLEIDUNG  RECN :ARTIKELB
   (OUT):   SETN :(MIN-BES  RLMN :          ITMN :          RECA :000001SO  SPP1 :ADMIN
   (OUT):   SPP2 :    KLEI  SPP3 :          SUBS :ADMIN
***(IN ): SH RECORD,L=6
   (OUT): RECORD - AREA      :000001
***(IN ): PROC ULI
***(IN ): FETCH DUPLICATE ARTIKELBESCHR
***(IN ): E
***(IN ): SH RECORD,LNG=6
***(IN ): END
***(IN ): DO ULI,R=4
***(IN ): FETCH DUPLICATE ARTIKELBESCHR
***(IN ): E
***(IN ): SH RECORD,LNG=6
   (OUT): RECORD - AREA      :000002
***(IN ): FETCH DUPLICATE ARTIKELBESCHR
***(IN ): E
***(IN ): SH RECORD,LNG=6
   (OUT): RECORD - AREA      :000003
***(IN ): FETCH DUPLICATE ARTIKELBESCHR
***(IN ): E
***(IN ): SH RECORD,LNG=6
   (OUT): RECORD - AREA      :000004
***(IN ): FETCH DUPLICATE ARTIKELBESCHR
***(IN ): E
***(IN ): SH RECORD,LNG=6
   (OUT): RECORD - AREA      :000005
```

```
***(IN ): DELETE ULI
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH2 ..  FOPT :DUPLIC..  SOPT :          UINF :KLEIDUNG  RECN :ARTIKELB
   (OUT):   SETN :(MIN-BES  RLMN :          ITMN :          RECA :OOOOO5SO  SPP1 :ADMIN
   (OUT):   SPP2 :    KLEI  SPP3 :          SUBS :ADMIN
```

FIND/FETCH (Format 3):

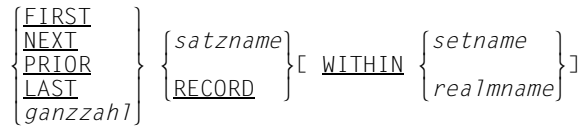
```
DUPLICATE WITHIN { satzname } [ USING satzelementname,...][ { IN } satzname]
                 { setname  }                              { OF }
```

Der Zusatz „IN *satzname“* ist erforderlich bei „DUPLICATE WITHIN *setname* USING *satzelementname*,...“, damit CALL-DML die Felder richtig identifizieren kann.

Beispiel

```
***(IN ): REMARK *** BEISPIEL FETCH 3 ***
***(IN ): FETCH 2 BESTELLUNG
***(IN ): E
***(IN ): SH RECORD
   (OUT): RECORD - AREA      :799E8201
***(IN ): FETCH DUPLICATE WITHIN ABGEGEBENE-BEST USING BEST-MONAT IN BESTELLUNG
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH3 ..  FOPT :SETITM..  SOPT :          UINF :BESTELLR  RECN :BESTELLU
   (OUT):   SETN :ABGEGEBE  RLMN :          ITMN :(BEST-MO  RECA :854.8201  SPP1 :ADMIN
   (OUT):   SPP2 :....KLEI  SPP3 :          SUBS :ADMIN
***(IN ): SH RECORD
   (OUT): RECORD - AREA      :854.8201
```

FIND/FETCH (Format 4):



Beispiel

```
***(IN ): REMARK *** BEISPIEL FETCH 4 ***
***(IN ): FETCH FIRST LIEFERANT
***(IN ): E
***(IN ): SH RECORD
   (OUT): RECORD - AREA      :00001MON
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH4 ..  FOPT :RECFST..  SOPT :          UINF :BESTELLR  RECN :LIEFERAN
   (OUT):   SETN :ABGEGEBE  RLMN :          ITMN :(BEST-MO  RECA :00001MON  SPP1 :ADMIN
   (OUT):   SPP2 :....KLEI  SPP3 :          SUBS :ADMIN
***(IN ): FETCH 2 BESTELLUNG WITHIN ABGEGEBENE-BEST
***(IN ): E
***(IN ): SH RECORD
   (OUT): RECORD - AREA      :854.8201
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH4 ..  FOPT :SETSPC..  SOPT :          UINF :BESTELLR  RECN :BESTELLU
   (OUT):   SETN :ABGEGEBE  RLMN :          ITMN :(BEST-MO  RECA :854.8201  SPP1 :ADMIN
   (OUT):   SPP2 :....KLEI  SPP3 :          SUBS :ADMIN
```

FIND/FETCH (Format 5):

```
CURRENT[ satzname][ WITHIN { setname   }]
                           { realmname }
```

Beispiel

```
***(IN ): REMARK *** BEISPIEL FETCH 5 ***
***(IN ): FIND FIRST ARTIKEL
***(IN ): E
***(IN ): FIND NEXT ARTIKEL RETAINING CURRENCY FOR LIEFERBARE-ARTIKEL
***(IN ): E
***(IN ): FETCH CURRENT ARTIKEL RETAINING CURRENCY FOR LIEFERBARE-ARTIKEL
***(IN ): E
***(IN ): SH RECORD,L=80
   (OUT): RECORD - AREA      :
   (OUT): 00000110SOMMERKLEID MIT JACKE                  23010738..................rn...
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH5 ..  FOPT :RECNAM..  SOPT :RET       UINF :KLEIDUNG  RECN :ARTIKEL
   (OUT):   SETN :ABGEGEBE  RLMN :          ITMN :(BEST-MO  RECA :00000110  SPP1 :      ST
   (OUT):   SPP2 :....KLEI  SPP3 :          SUBS :ADMIN
***(IN ): FETCH CURRENT ARTIKEL WITHIN LIEFERBARE-ARTIKEL
***(IN ): E
***(IN ): SH RECORD,L=80
   (OUT): RECORD - AREA      :
   (OUT): 00000110SOMMERKLEID MIT JACKE                  23010736..................rn...
```

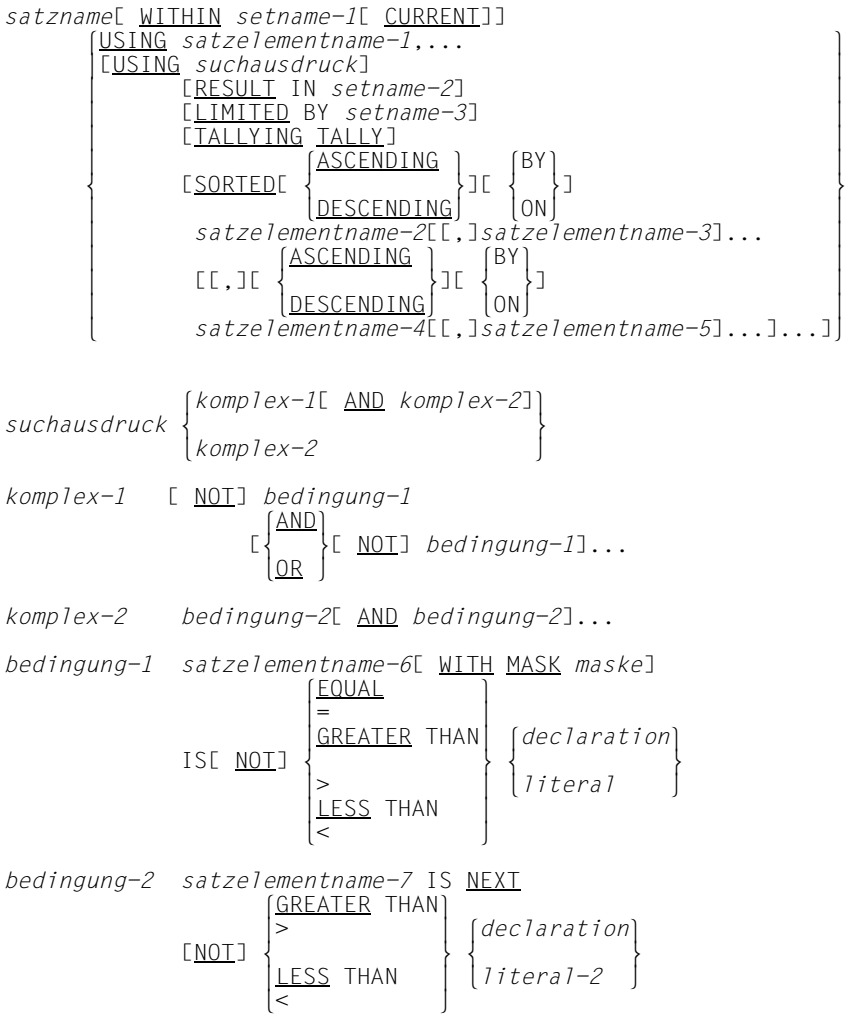
FIND/FETCH (Format 6):

OWNER WITHIN *setname*

Beispiel

```
***(IN ): REMARK *** BEISPIEL FETCH 6 ***
***(IN ): FIND 1 BESTELLUNG
***(IN ): E
***(IN ): FETCH CURRENT BESTELLUNG WITHIN ABGEGEBENE-BEST
***(IN ): E
***(IN ): SH RECORD
   (OUT): RECORD - AREA      :000C8201
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH5 ..  FOPT :RECSET..  SOPT :          UINF :BESTELLR  RECN :BESTELLU
   (OUT):   SETN :ABGEGEBE  RLMN :          ITMN :(BEST-MO  RECA :000C8201  SPP1 :      ST
   (OUT):   SPP2 :....KLEI  SPP3 :          SUBS :ADMIN
***(IN ): FETCH OWNER WITHIN ABGEGEBENE-BEST
***(IN ): E
***(IN ): SH RECORD
   (OUT): RECORD - AREA      :00001MON
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH6 ..  FOPT :RECSET..  SOPT :          UINF :BESTELLR  RECN :BESTELLU
   (OUT):   SETN :ABGEGEBE  RLMN :          ITMN :(BEST-MO  RECA :00001MON  SPP1 :      ST
   (OUT):   SPP2 :....KLEI  SPP3 :          SUBS :ADMIN
```

FIND/FETCH (Format 7):



satzelementname-2...5
: entspricht einem Satzelement in der Datenbank

declaration
: entspricht einem DECLARE-Feld

Wenn Sie DECLARE-Felder als Vergleichsfelder benutzen, müssen Sie beachten, wie das MOVE-Kommando den Inhalt der Felder eingetragen hat (siehe Seite 328).

Für Maskenfelder benutzen Sie DECLARE-Felder. Diese DECLARE-Felder müssen mit der Länge der Felder in der Datenbank übereinstimmen.
Jedes Byte muss als signifikant (X'F1') oder nicht signifikant (X'F0') gekennzeichnet sein.

Beispiele

```
***(IN ): REMARK *** BEISPIEL FETCH 7 ***
***(IN ): REMARK **FETCH-7-SATZELEMENTNAMEN**

***(IN ): S RECA=C' '
***(IN ): M RECA,C'    8201'
***(IN ): FETCH BESTELLUNG USING BEST-MONAT,BEST-JAHR
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH7A..  FOPT :RECITM..  SOPT :          UINF :BESTELLR  RECN :BESTELLU
   (OUT):   SETN :ABGEGEBE  RLMN :          ITMN :(BEST-MO  RECA :000C8201  SPP1 :      ST
   (OUT):   SPP2 :....KLEI  SPP3 :          SUBS :ADMIN
***(IN ): SH RECORD,L=10
   (OUT): RECORD - AREA      :000C82011E
***(IN ): PROC ULI
***(IN ): FETCH DUPLICATE WITHIN BESTELLUNG USING BEST-MONAT,BEST-JAHR
***(IN ): E
***(IN ): SH RECORD,L=10
***(IN ): END
***(IN ): DO ULI,R=3
***(IN ): FETCH DUPLICATE WITHIN BESTELLUNG USING BEST-MONAT,BEST-JAHR
***(IN ): E
***(IN ): SH RECORD,L=10
   (OUT): RECORD - AREA      :799E82011.
***(IN ): FETCH DUPLICATE WITHIN BESTELLUNG USING BEST-MONAT,BEST-JAHR
***(IN ): E
***(IN ): SH RECORD,L=10
   (OUT): RECORD - AREA      :854.82010C
***(IN ): FETCH DUPLICATE WITHIN BESTELLUNG USING BEST-MONAT,BEST-JAHR
***(IN ): E
***(IN ): SH RECORD,L=10
   (OUT): RECORD - AREA      :785D82010C
***(IN ): DEL ULI


***(IN ): REMARK **FETCH-7-SUCHAUSDRUCK**

***(IN ): RUN PO
***(IN ): DCL PARAMS,L=41
***(IN ): M PARAMS,*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN ): SH PARAMS,L=41
   (OUT): PARAMS             :*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN ): DEL PARAMS
***(IN ): SET FOPT=C' '
***(IN ): SET SOPT=C' '
***(IN ): SET RECN=C' '
```

```
***(IN ): SET ITMN=C' '
***(IN ): SET RECA=C' '
***(IN ): S SPP1=C' '
***(IN ): S SPP2=C' '
***(IN ): S RLMN=C' '
***(IN ): FIND 1 ARTIKELBESCHR
***(IN ): E
***(IN ): FETCH ARTIKEL WITHIN BESTELLANGABEN USING GROESSE IS > 40
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :FTCH7A..  FOPT :SELSEX..  SOPT :          UINF :KLEIDUNG  RECN :ARTIKEL
   (OUT):   SETN :BESTELLA  RLMN :          ITMN :0 GROESS  RECA :00000110  SPP1 :
   (OUT):   SPP2 :....      SPP3 :          SUBS :ADMIN
***(IN ): SH RECA,L=80
   (OUT): RECA  CONTAINS     :
   (OUT): 00000110SOMMERKLEID MIT JACKE                  23010742..................rn...
***(IN ): PROC ULI
***(IN ): FETCH DUPLICATE WITHIN BESTELLANGABEN
***(IN ): E
***(IN ): SH RECA,L=80
***(IN ): END
***(IN ): DO ULI,R=3,COND=RCODE EQ C'00000'
***(IN ): FETCH DUPLICATE WITHIN BESTELLANGABEN
***(IN ): E
***(IN ): SH RECA,L=80
   (OUT): RECA  CONTAINS     :
   (OUT): 00000110SOMMERKLEID MIT JACKE                  23010744..................rn...
***(IN ): FETCH DUPLICATE WITHIN BESTELLANGABEN
***(IN ): E
***(IN ): SH RECA,L=80
   (OUT): RECA  CONTAINS     :
   (OUT): 00000110SOMMERKLEID MIT JACKE                  23010746.. .................q...
***(IN ): FETCH DUPLICATE WITHIN BESTELLANGABEN
***(IN ): E
***(IN ): SH RECA,L=80
   (OUT): RECA  CONTAINS     :
   (OUT): 00000110SOMMERKLEID MIT JACKE                  23010748.. .................q...
```

Den Zugriff auf den vorigen oder nächsten Satz (COBOL-Parameter OR PRIOR/NEXT), wenn es für die angegeben Werte keinen Satz in der Datenbank gibt, können Sie dadurch realisieren, dass Sie zunächst die FIND/FETCH-Anweisung eingeben und dann vor der Ausführung noch den Parameter FOPT mit RECITP/SECITN oder ...ITN versorgen (z.B. FETCH BESTELLUNG USING BEST-MONAT,BEST-JAHR; MOVE FOPT,RECITN;E).

FINISH

```
FINISH[ WITH CANCEL]
```

FREE

```
FREE[ ALL]
```

GET

```
GET [{satzname      }][ {IN} satzname]]
    {feldname,...   }  {OF}
```

IF

IF (Format 1):

```
IF[ setname-1] {OWNER  }
               {MEMBER }
               {TENANT }
```

IF (Format 2):

```
IF setname-2 IS EMPTY
```

KEEP

```
KEEP
```

MODIFY

```
MODIFY {satzname                                  }
       {                                          }
       {satzelementname,... {IN}satzname          }
       {                    {OF}                  }

       [{INCLUDING} {ALL             } MEMBERSHIP]
        {ONLY     } {setname-1,...   }

       [RETAINING CURRENCY FOR {SETS          }]
                               {setname-1,... }
```

Der Zusatz „IN *satzname“* ist erforderlich, damit CALL-DML die Felder richtig identifizieren kann.

Beispiele

```
***(IN ): REMARK *** BEISPIEL MODIFY-SATZELEMENT***

***(IN ): FETCH 1 LIEFERANT
***(IN ): E
***(IN ): SH RECA,L=104
   (OUT): RECA  CONTAINS     :
   (OUT): 00001MONA MODEHAUS                 7500KARLSRUHE 1                   AUGUSTENSTR
   (OUT): ASSE              00100
***(IN ): M RECA,999,D=99
***(IN ): MODIFY LIEFER-HAUSNR IN LIEFERANT
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :MODIF2..  FOPT :CORUNT..  SOPT :          UINF :          RECN :LIEFERAN
   (OUT):   SETN :BESTELLA  RLMN :          ITMN :(LIEFER-  RECA :00001MON  SPP1 :ADMIN
   (OUT):   SPP2 :....4     SPP3 :          SUBS :ADMIN
***(IN ): SH RECA,L=104
   (OUT): RECA  CONTAINS     :
   (OUT): 00001MONA MODEHAUS                 7500KARLSRUHE 1                   AUGUSTENSTR
   (OUT): ASSE              99900
***(IN ): M RECA,001,D=99
***(IN ): MODIFY LIEFER-HAUSNR
***(IN ): E


***(IN ): REMARK *** BEISPIEL MODIFY-OWNERZUORDNUNG***

***(IN ): RUN PO
***(IN ): DCL PARAMS,L=41
***(IN ): M PARAMS,*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN ): SH PARAMS,L=41
   (OUT): PARAMS             :*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN ): DEL PARAMS
***(IN ): SET FOPT=C' '
***(IN ): SET SOPT=C' '
***(IN ): SET RECN=C' '
***(IN ): SET ITMN=C' '
***(IN ): SET RECA=C' '
***(IN ): S SPP1=C' '
***(IN ): S SPP2=C' '
***(IN ): S RLMN=C' '
***(IN ): FIND 1 BESTELLUNG
***(IN ): E
***(IN ): FETCH OWNER WITHIN ABGEGEBENE-BEST
***(IN ): E
***(IN ): SH RECA,L=130
   (OUT): RECA  CONTAINS     :
   (OUT): 00001MONA MODEHAUS                 7500KARLSRUHE 1                   AUGUSTENSTR
   (OUT): ASSE              001000721609422................
***(IN ): M RECA,00002AUGUSTINERBRAEU
***(IN ): FIND ANY LIEFERANT
```

```
***(IN ): E
***(IN ): FIND 1 BESTELLUNG RETAINING CURRENCY FOR MULTIPLE
***(IN ): E
***(IN ): MODIFY BESTELLUNG ONLY ABGEGEBENE-BEST MEMBERSHIP
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :MODIF1..  FOPT :ONLSET..  SOPT :          UINF :          RECN :BESTELLU
   (OUT):   SETN :(ABGEGEB  RLMN :          ITMN :          RECA :00002AUG  SPP1 :MULTIPLE
   (OUT):   SPP2 :....      SPP3 :          SUBS :ADMIN
***(IN ): FETCH OWNER WITHIN ABGEGEBENE-BEST
***(IN ): E
***(IN ): SH RECA,L=130
   (OUT): RECA  CONTAINS     :
   (OUT): 00002AUGUSTINERBRAEU              8000MUENCHEN 2                    MARSSTRASSE
   (OUT):                    77 000899128741.............
***(IN ): FIND 1 LIEFERANT
***(IN ): E
***(IN ): FIND 1 BESTELLUNG RETAINING MULTIPLE
***(IN ): E
```

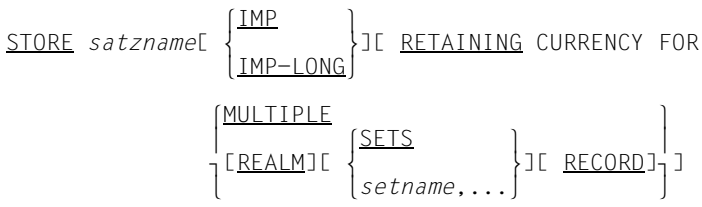
READY

```
READY[ realmname,...]
      [USAGE-MODE IS[ {EXCLUSIVE } ]{RETRIEVAL}]
                     {PROTECTED }  {UPDATE   }
```

Beispiel

```
***(IN ): SUBS IS ADMIN
***(IN ): M USERGROUP,CL8'VERKAUF'
***(IN ): M USERNAME,CL24'MAYER'
***(IN ): M PASSWORD,CL48'HUGO'
***(IN ): READY USAGE-MODE IS EXCLUSIVE UPDATE
***(IN ): EX
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :READYC..  FOPT :ALLEUP..  SOPT :          UINF :          RECN :ARTIKEL
   (OUT):   SETN :          RLMN :          ITMN :          RECA :........  SPP1 :ADMIN
   (OUT):   SPP2 :VERKAUF   SPP3 :          SUBS :ADMIN
```

STORE



Den Zusatz IMP bzw. IMP-LONG müssen Sie in folgenden Fällen angeben:

– Die Satzart *satzname* kann in mehrere Realms gespeichert werden.

  Wenn die Satzart *satzname* **nicht** mit der DDL-Klausel LOCATION MODE definiert wurde (siehe Handbuch „Entwerfen und Definieren“), können Sie wahlweise IMP oder IMP-LONG angeben.
  Außerdem müssen Sie das vordefinierte Feld IMP-AREA-ID mit dem Namen des Realm versorgen, in den der Satz gespeichert werden soll.

  Eine Ausnahme von obiger Regel bilden verteilbare Listen:
  Beim Abspeichern eines Satzes, der Member einer verteilbaren Liste ist, wird im DBH eine benötigte freie Seite in dem bevorzugten Realm (Preferred-Realm) gesucht. Die Angabe eines Realms im Feld IMP-AREA-ID ist in diesem Fall bedeutungslos.

– Die Satzart *satzname* wurde mit LOCATION DIRECT bzw. LOCATION DIRECT-LONG definiert (siehe Handbuch „Entwerfen und Definieren“).

  IMP geben Sie an im Fall LOCATION DIRECT. IMP-LONG geben Sie an im Fall LOCATION DIRECT-LONG.
  Außerdem müssen Sie vor dem Speichern das vordefinierte Feld IMP-DB-KEY bzw. IMP-DB-KEY-LONG mit dem gewünschten Database-Key-Wert oder binär 0 versorgen.

Beispiele

```
***(IN ): REMARK *** BEISPIEL STORE-RECORD ***
***(IN ): RUN PO
***(IN ): DCL PARAMS,L=41
***(IN ): M PARAMS,*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN ): SH PARAMS,L=41
   (OUT): PARAMS             :*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN ): DEL PARAMS
***(IN ): SET FOPT=C' '
***(IN ): SET SOPT=C' '
***(IN ): SET RECN=C' '
***(IN ): SET ITMN=C' '
***(IN ): SET RECA=C' '
***(IN ): S SPP1=C' '
***(IN ): S SPP2=C' '
***(IN ): S RLMN=C' '
```

```
***(IN ): FETCH LAST ARTIKEL
***(IN ): E
***(IN ): S RECA=C' '
***(IN ): M UINF,LEBENSMITTEL
***(IN ): M RECA,C'001144 ODR.HUBERSVITAMINSAFT'
***(IN ): STORE ARTIKEL
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :STORE1..  FOPT :IMPDAT..  SOPT :          UINF :LEBENSMI  RECN :ARTIKEL
   (OUT):   SETN :(ABGEGEB  RLMN :          ITMN :          RECA :001144 0  SPP1 :
   (OUT):   SPP2 :...r      SPP3 :          SUBS :ADMIN
***(IN ): FETCH LAST ARTIKEL
***(IN ): E
***(IN ): SH RECA,L=80
   (OUT): RECA  CONTAINS     :
   (OUT): 001144 ODR.HUBERSVITAMINSAFT
***(IN ): ERASE ARTIKEL
***(IN ): E


***(IN ): REMARK *** BEISPIEL STORE-SET-OCCURRENCE ***

***(IN ): S RECA=C' '
***(IN ): M RECA,000003
***(IN ): FETCH ARTIKELBESCHR USING ART-NR
***(IN ): E
***(IN ): S RECA=C' '
***(IN ): M RECA,00002AUGUSTINER
***(IN ): FETCH LIEFERANT USING LIEFER-NR
***(IN ): E
***(IN ): S RECA=C' '
***(IN ): M RECA,C'901145 OMUELLER SCHLANKHEITSKOST'
***(IN ): STORE ARTIKEL
***(IN ): E
***(IN ): SH PARAM
   (OUT): DB-PARAMS (FIRST 8B)
   (OUT):   FCOD :STORE1..  FOPT :RECNAM..  SOPT :          UINF :LEBENSMI  RECN :ARTIKEL
   (OUT):   SETN :(ABGEGEB  RLMN :          ITMN :(LIEFER-  RECA :901145 0  SPP1 :
   (OUT):   SPP2 :...r      SPP3 :          SUBS :ADMIN
***(IN ): FETCH LAST ARTIKEL
***(IN ): E
***(IN ): SH RECA,L=80
   (OUT): RECA  CONTAINS     :
   (OUT): 901145 OMUELLER SCHLANKHEITSKOST
```

9.4 Ablauf des Programms DMLTEST

In diesem Abschnitt finden Sie Informationen zu folgenden Themen:

– Unterbrechungen, die während eines Programmlaufs auftreten können

– Kommunikation von DMLTEST mit einer oder mehreren Datenbanken

Vor der Ausführung von DMLTEST müssen das Subschema, mit dem Sie arbeiten wollen, mit dem Dienstprogramm BCALLSI vorbereitet haben (siehe Handbuch „Aufbauen und Umstrukturieren“).

9.4.1 Unterbrechungen

Ein Unterbrechungszustand wird erreicht durch:

– logische und syntaktische Fehler in einem Kommando
– das Erreichen des TRACE-Limit
– das WAIT-Kommando
– einen Bildschirmüberlauf

Wenn Sie im Dialog arbeiten, wird eine Eingabe vom Bildschirm erwartet.

Der Unterbrechungszustand wird durch die Meldung

```
STATUS IS BREAK
```

gekennzeichnet.

Mit folgenden DMLTEST-Kommandos können Sie auf eine Unterbrechung reagieren:

NEXT wenn Sie die Bearbeitung eines Kommandos abbrechen wollen. Das nächste Kommando der aktuellen Kommandofolge wird gestartet.

CONTINUE wenn Sie die Verarbeitung einfach fortsetzen wollen

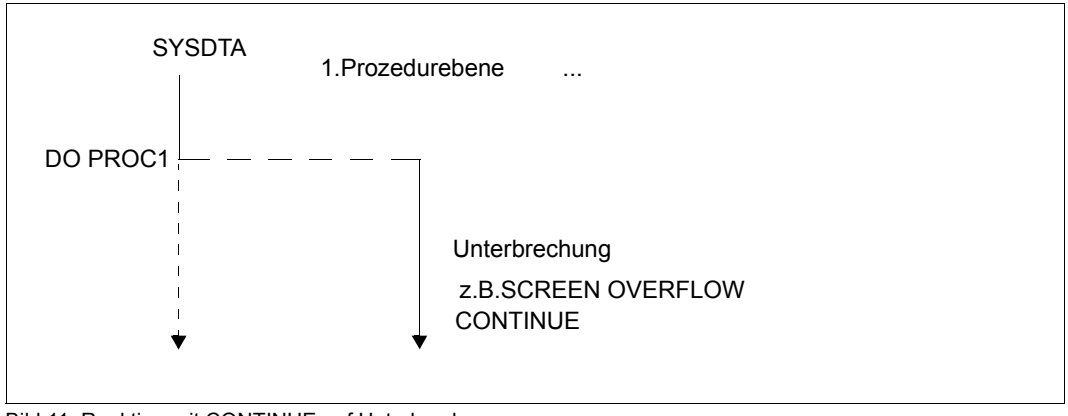


Bild 11: Reaktion mit CONTINUE auf Unterbrechungen

ESCAPE wenn Sie die Bearbeitung aller begonnenen Prozeduren oder Kommandofolgen abbrechen wollen. Es wird von SYSDTA das nächste Kommando erwartet.

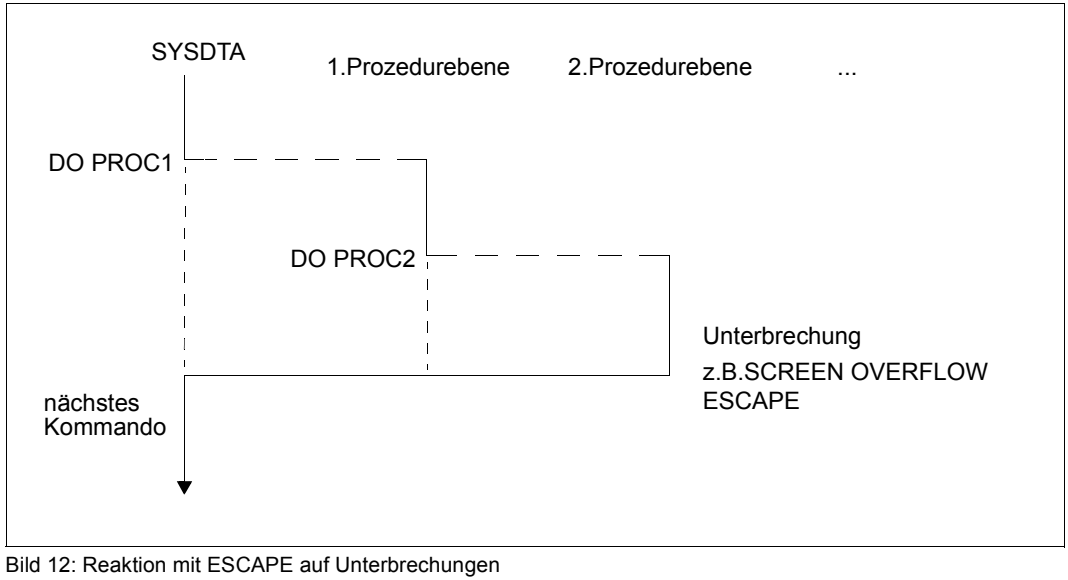


Bild 12: Reaktion mit ESCAPE auf Unterbrechungen

LEAVE wenn Sie die Bearbeitung der aktuellen Prozedur oder Kommandofolge abbrechen wollen und die Steuerung an die nächsthöhere Prozedur, Kommandofolge bzw. den Bildschirm geben wollen.

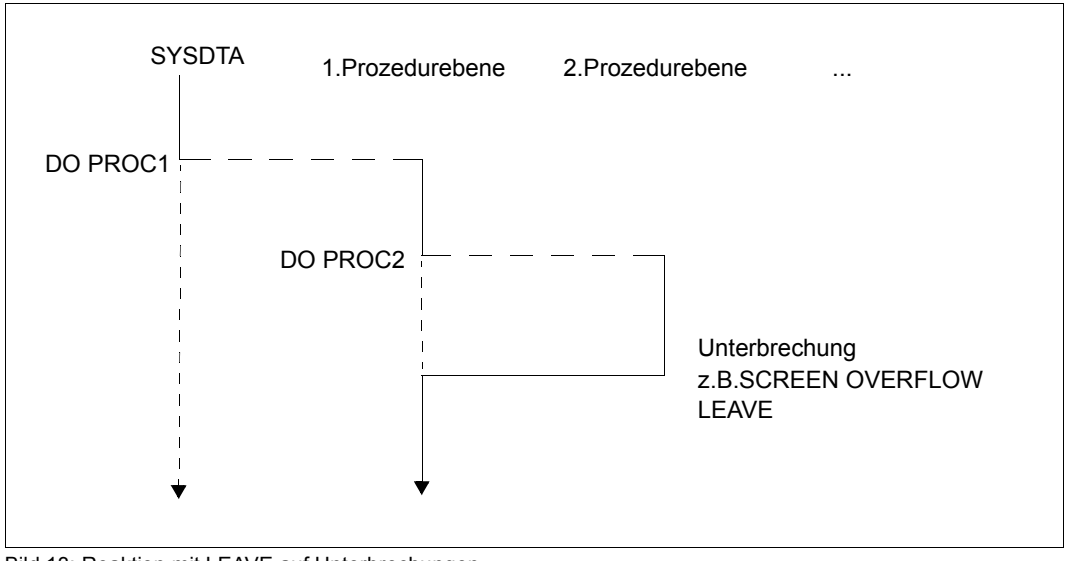


Bild 13: Reaktion mit LEAVE auf Unterbrechungen

Die Unterbrechung wird durch CONTINUE, ESCAPE oder NEXT beendet. Bei LEAVE müssen Sie noch zusätzlich eines dieser drei Kommandos eingeben, wenn Sie das Beenden der Unterbrechung wünschen.

9.4.2 Die Kommunikation mit einer oder mehreren Datenbanken

Um mit einer Datenbank zu arbeiten, müssen Sie zuerst folgende Angaben machen:

– die DBH-Variante auswählen (DBH-Kommando)

– die Sprache auswählen (LANGUAGE-Kommando)

– das Subschema auswählen
(SET SUBS=*subschemaname* oder SUBSCHEMA IS *subschemaname*)

Danach können Sie eine READY-Anweisung ausführen.

Für jede DML-Anweisung gilt:
Haben Sie die Parameterleiste mit Werten versorgt, bleiben die Felder so lange unverändert, bis Sie sie mit neuen Werten überschreiben. Satzbereich und Benutzerinformationsbereich werden auch von den DML-Anweisungen überschrieben.
Sie können das EXECUTE-Kommando mit den gleichen Parameterwerten so oft wiederholen, wie Sie wollen.

Satzbereich und Benutzerinformationsbereich weisen auf die zuletzt ausgeführte Aufgabe. Die Kommandos SHOW, PRINT oder DISPLAY zeigen Ihnen die Ergebnisse.

Multi-DB

Arbeiten Sie mit DMLTEST in mehreren Datenbanken, müssen Sie darauf achten, dass jede der Datenbanken symbolisch ansprechbar bleibt.

Die Identifikation einer Datenbank erfolgt über den Subschemanamen. Sie geben mit SET SUBS=*subschemaname* bzw. SUBS IS *subschemaname* die entsprechende Information an DMLTEST. Die ersten 6 Zeichen werden als symbolischer Name der Datenbank verwendet. Sie müssen aber die volle Länge (30 Zeichen) angeben.
Verschiedene Subschemanamen müssen in den ersten 6 Zeichen eindeutig sein.

Bei KDBS sorgt der Umsetzer für die Auswahl der richtigen Datenbank.

9.5 Fehlermeldungen

Tritt eine Unterbrechung auf Grund eines Fehlers auf, bekommen Sie eine Meldung. Reicht der allgemeine Fehlertext nicht aus, geben Sie ein HELP-Kommando ein.

Beispiel

```
***(IN ): DEF RECORD,NACHNAME,D=10,L=15
   (OUT): DMLTEST: USED NAME. STATUS IS NORMAL.
***(IN ): HELP
   (OUT): LAST INPUT         :DEF RECORD,NACHNAME,D=10,L=15
   (OUT): FROM SYSDTA         STATUS IS NORMAL
   (OUT): ERROR DESCRIPTION  :NAME IST SCHON FUER EINE DEFINITION VERGEBEN!
```

Die Bezeichnungen in den allgemeinen Fehlertexten haben folgende Bedeutungen:

PARAM-NAME: Parameter oder Feld

DB-NAME: Feld im Subschema

ELEMENT: Feld nicht identifizierbar

Zusammenfassung der allgemeinen Fehlertexte

```
COMMAND NOT FOUND
MISSING NECESSARY OPERAND
SYNTAX ERROR
OPERAND ERROR
OPERAND-NAME NOT FOUND
TOO MANY OPERANDS
INTERNAL BUFFER OVERFLOW
TOO LARGE DISTANCE
PARAM-NAME NOT FOUND
COMMAND ILLEGAL IN CONTEXT
EDT NOT AVAILABLE
DSCEXT WAS CALLED BY UDS[1]
ERROR ON RUN-FILE
NOT READY EXECUTED
DB-NAME UNKNOWN
DB-ERROR STATUS
ELEMENT NOT ALLOWED
NUMERICAL ERROR
USED NAME
WARNING
```

1 Wenn der Benutzerinformationsbereich zerstört ist, müssen Sie ihn selbst wieder rekonstruieren (siehe Seite 124).

10 Anhang

10.1 Statuscodes

Statuscodes der DML

Statuscode als Hinweis

001 Bei einem FIND/FETCH Format 1 oder 7 mit OR PRIOR/OR NEXT-Angabe: Es wurde kein Satz gefunden, der mit den vorgegebenen Werten übereinstimmt. Der nächste Satz in der Sortierfolge wurde zur Verfügung gestellt.

Statuscodes mit Fortschrittshinweisen der Online-Utility

010 RELOCATE DML: Quell- und Zielpegel sind gleich. Die Verlagerung ist beendet.

REORGPPP DML: Ende der Realms erreicht. Die Reorganisation ist beendet.

011 RELOCATE DML: Quell- und Zielpegel sind 0 bei INITIALIZE=*NO.

REORGPPP DML: Die derzeitige Seitenzahl ist 0 bei INITIALIZE=*NO.

Bei einer versuchten Fortsetzung von Verlagerungen mit INITIALIZE=*NO wird festgestellt, dass keine Informationen mehr vorliegen, z.B. weil die Datenbank zwischenzeitlich ausgehängt worden ist oder ein neuer Sessionabschnitt begonnen wurde.

012 RELOCATE DML: Beim Lesen einer Quellseite ist ein Sperrkonflikt mit einer parallelen Transaktion aufgetreten.

REORGPPP DML: Beim Lesen einer Seite ist ein Sperrkonflikt mit einer parallelen Transaktion aufgetreten.

013 Beim Lesen einer Zielseite ist ein Sperrkonflikt mit einer parallelen Transaktion aufgetreten.

Statuscodes zur Datenkonsistenz

018 Deadlock-Zustand (gegenseitiges Sperren mehrerer Transaktionen auf UDS/SQL-Betriebsmitteln);
FINISH WITH CANCEL wird ausgeführt. Es ist sinnvoll, die Transaktionen zu wiederholen (begrenzt).
Für UDS-D:
Im UDS/SQL-Betrieb ohne openUTM erfolgt die globale Deadlock-Erkennung über eine Zeitüberwachung (PP DEADTIME) von Wartesituationen. Nach Ablauf dieses Zeitlimits wird der Statuscode 018 angezeigt, auch wenn u. U. kein wirklicher Deadlock vorliegt.

020 FIND/FETCH (nur CALL-DML)
Eine Seite, auf die zugegriffen werden soll, ist von einer anderen Transaktion gesperrt.

Statuscodes zur Satz-Wiedergewinnung

021 Das Ende einer Satzart, eines Sets oder Realms wurde erreicht.

FIND/FETCH Formate 2 (DUPLICATE) und 3 (USING):
Es kann kein Satz mit gleichen Werten wie der entsprechende CRR bzw. CRS gefunden werden.

FIND/FETCH Format 3 (ohne USING):
Das Ende der Treffermenge wurde erreicht.

FIND/FETCH Format 4:
Kein nächster (NEXT) oder vorhergehender (PRIOR) Satz kann gefunden werden oder
ganzzahl bzw. *name* enthält einen Wert, der keinen Satz innerhalb des Realms/der Satzart/der Set-Occurrence adressiert.

022 Die Transaktion versucht einen Realm zu eröffnen, der für UPDATE und RETRIEVAL gesperrt ist. Mögliche Ursachen für diese Sperre sind auf

Datenbankebene:

- Die Datenbank wurde vom Datenbankadministrator via DAL-ACCESS-Kommando gesperrt.
- Das DBDIR der Datenbank ist gesperrt (siehe „Realm-Ebene“).

Realm-Ebene:

- Der Realm wurde im Rahmen einer Datenbankrestrukturierung aus der Datenbank gestrichen.
- Der Realm wurde vom Datenbankadministrator oder von der UDS/SQL-Fehlerbehandlung abgeschaltet.
- Der Realm wurde vom Datenbankadministrator via DAL-ACCESS-Kommando gesperrt.

023 Nur bei SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER: Es kann keine den Setauswahlkriterien genügende Set-Occurrence gefunden werden.

024 Es kann kein dem Satzauswahlausdruck genügender Satz gefunden werden.

FIND/FETCH Format 1:
Der Database Key liefert aus einem der folgenden Gründe keinen Treffer:
- Seine Satzartnummer entspricht nicht der explizit angegebenen Satzart.
- Sein Wert liegt zwar innerhalb der Grenzen seiner DBTT, es existiert aber in der Datenbank kein zugehöriger Satz.

FIND/FETCH Formate 2 (ANY) und 7:
Es kann kein Satz gefunden werden, der den initiierten Datenelementen bzw. dem Suchausdruck entspricht.

FIND/FETCH Format 4:
Es kann kein Satz innerhalb der angegebenen Satzart, des Realms oder der Set-Occurrence gefunden werden.

027 Die Subskribierung des angegebenen Feldnamens liegt nicht innerhalb des durch die OCCURS-Klausel im Subschema festgelegten Bereiches.

028 Der angegebene Database Key enthält eine ungültige Satzartnummer oder eine Satzfolgenummer, die außerhalb der Grenzen seiner DBTT liegt.

029 FIND/FETCH Format 4 und 5:
Der Current des Realms bzw. Sets hat nicht die in der Anweisung angegebene Satzart.

Statuscodes zu Currency-Indikatoren

031 Der Current des Realms, des Sets oder der Satzart ist nicht bekannt.

FIND/FETCH Format 3:
Der Current of Set ist Owner und nicht Member des angegebenen Sets oder der angegebene Setname unterscheidet sich von dem im vorausgegangenen FIND7 angegebenen Setnamen.

FIND/FETCH Format 6 und Format 7:
Der Owner wurde gelöscht.

IF Format 2:
Der CRS wurde gelöscht oder aus dem angegebenen Set ausgehängt.

032 Der Current of Rununit ist nicht bekannt oder wurde gelöscht.

033 Der Current of Rununit hat nicht die in der Anweisung angegebene Satzart.

Statuscodes zur Namensgebung

042 Satzart, Set oder Realm sind nicht im aufgerufenen Subschema definiert oder

ein Feld, das Teil eines ASC-, DESC-, CALC-Keys ist, ist nicht im Subschema definiert oder

nach einer Subschemaänderung wurde das Anwenderprogramm nicht neu übersetzt (COBOL-DML) oder der BCALLSI-Lauf vergessen (CALL-DML) oder

Fehler an der BIB-Schnittstelle (siehe Statuscode 103) oder

bei einer Online-Utility wurde ein Realm angegeben, in dem keine Aktivitäten zulässig sind.

043 STORE und FIND/FETCH Format 2:
Das AREA-ID-Datenelement enthält den Namen eines Realm, der nicht in der DDL-WITHIN-Klausel angegeben ist oder nicht zum aufgerufenen Subschema gehört oder
bei SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER und Ownersatzart = LOCATION MODE IS CALC: Das AREA-ID-Datenfeld des Ownersatzes enthält den Namen eines Realm, der nicht in der DDL-WITHIN-Klausel angegeben ist oder nicht zum aufgerufenen Subschema gehört.

044 IF:
Die Angabe eines dynamischen Sets ist nicht erlaubt.

Statuscodes zur Eindeutigkeit von Schlüsseln

051 Doppeltes Auftreten von Schlüsselwerten in der Datenbank. Das bedeutet, dass die Ausführung einer DML-Anweisung einer DUPLICATES ARE NOT ALLOWED-Angabe in einer ORDER IS SORTED BY DEFINED KEYS-Klausel oder SEARCH-KEY-Klausel eines Sets, in dem der betroffene Satz Member ist, oder der LOCATION MODE IS CALC-Klausel oder SEARCH-KEY-Klausel des betroffenen Satzes widersprechen würde.

Statuscodes zu Satzeigenschaften

071 FIND/FETCH Format 2 (DUPLICATE), 3, 4 und 5:
Der Aufsetzpunkt der DML-Anweisung (CRR, CRA bzw. CRS) wurde gelöscht oder aus der aktuellen Set-Occurrence ausgehängt. Im Falle der Abarbeitung einer Treffermenge (FIND3 ohne USING) führen Änderungsoperationen der eigenen Transaktion nicht zum Verlust des Aufsetzpunktes, nur Änderungen fremder Transaktionen.

072 ERASE:
Der betroffene Satz ist Owner einer nicht leeren Set-Occurrence und kann daher mit der gewählten ERASE-Variante nicht gelöscht werden.

Statuscodes zur Set-Mitgliedschaft

081 CONNECT (setname):
Der CRU ist in einem der spezifizierten Sets bereits Member oder einer der spezifizierten Sets ist kein Member-Set des CRU.
CONNECT (ALL):
Der CRU ist in allen seinen Member-Sets bereits Member.
MODIFY (setname):
Einer der spezifizierten Sets ist kein Member-Set des CRU.
MODIFY (ALL):
Der CRU ist in keinem seiner Member-Sets Member.

082 DISCONNECT (setname):
Der CRU ist in einem der spezifizierten Sets MANDATORY Member oder einer der spezifizierten Sets ist kein Member-Set des CRU
DISCONNECT (FROM ALL):
Kein Member-Set des CRU ist OPTIONAL.

083 DISCONNECT (setname) und MODIFY (setname):
Der CRU ist in einem der spezifizierten Sets kein Member
DISCONNECT (FROM ALL):
Mindestens ein Member-Set des CRU ist OPTIONAL, aber in keinem dieser OPTIONAL Sets ist der CRU Member.

Statuscodes zum READY-Zustand

091 Ein Realm ist nicht im READY-Zustand (d.h. ein Realm wurde beim READY nicht explizit angegeben oder ist nicht Teil des aktuellen Subschemas) oder

bei einem ERASE PERMANENT/SELECTIVE/ALL wurden beim READY Realm-Namen explizit angegeben oder

die DBTT einer zu verlagernden Satzart bei der Online-Utility liegt in einem nicht eröffneten Realm.

092 In einer RETRIEVAL-Verarbeitungskette ist keine DML-Anweisung mit Update-Funktion zulässig
oder bei einem ERASE PERMANENT/SELECTIVE/ALL wurde die Verarbeitungskette nicht mit EXCLUSIVE UPDATE eröffnet
oder im Falle des P-Parameters PP TA-ACCESS=SHARED wird versucht, eine Verarbeitungskette in den Benutzungsarten PROTECTED oder EXCLUSIVE zu eröffnen.

093 Der Database Handler lässt die Verarbeitungskette nicht zu, da die betreffende Datenbank innerhalb der Transaktion bereits eröffnet ist („Second-READY“ innerhalb einer Verarbeitungskette).

099 (nur CALL-DML oder Online-Utility)
Beim Eröffnen einer Transaktion ist ein Realm von einer anderen Transaktion gesperrt.

Statuscodes für fehlerhafte DML-Anweisungen

101 FIND/FETCH Format 4:
Es wurde für den Eintrag *ganzzahl* oder *feldname* der Wert Null verwendet
oder
bei der Suche in einer nicht rückwärts verketteten CHAIN wurde ein negativer Wert verwendet.

FIND/FETCH Format 6:
setname darf keinen singulären Set bezeichnen.

FIND/FETCH Format 7:

- OR PRIOR oder OR NEXT konnte nicht durchgeführt werden, weil kein sortierter und indizierter Schüssel vorlag.
- „WITHIN *setname-1* USING *satzelementname-1*,...“ wurde angegeben.
  die Angabe eines dynamischen Set in *setname-1* ist nicht erlaubt.
- LIMITED BY *dynamischer-set* ... SORTED BY ... wurde angegeben.
  Das Sortieren des Durchschnitts einer Treffermenge und eines dynamischen Set ist nicht möglich.

– LIMITED BY *sortierter-dynamischer-set* ... wurde angegeben.
Die Durchschnittbildung einer Treffermenge mit einem sortierten dynamischen Set ist nicht möglich.

FINISH:
FINISH-Typ (mit oder ohne CANCEL) nicht identifizierbar.

102 SET, ACCEPT (Format 1):
Ein großer Database-Key-Wert (Database-Key-Wert mit einer REC-REF > 254 und/oder einer RSQ > 2^{24}-1) kann nicht in ein Feld des Typs USAGE IS DATABASE-KEY übertragen werden. Es ist ein Subschema zu verwenden, in dem nicht
SUBSCHEMA FORM IS OLD angegeben ist und das ab UDS/SQL V2.0 erzeugt wurde, zusätzlich muss das angegebene Feld vom Typ
USAGE IS DATABASE-KEY-LONG sein.

103 Fehler an der BIB-Schnittstelle.
Mögliche Ursachen: Falscher COBOL-Compiler oder falsches COBOL-Laufzeitsystem, Fehler im CALL-DML-Konverter, in IQS, in der Online-Utility, oder in einem Dienstprogramm, welches BIBs erzeugt, oder Fehler im Database Handler.

Statuscode zu System-Fehlern

113 Beim Zugriff auf eine Datenbankseite wurde ein schwerwiegender Fehler im Database Handler oder in der Datenbank entdeckt.

Statuscodes zu den UDS/SQL-Betriebsmitteln

122 Die Transaktion wurde vom DBH vorzeitig mit CANCEL beendet.
Mögliche Ursachen:

– RLOG-Datei zu klein oder zu oft gesplittet.

– UDS/SQL-Puffer zu klein, PP BUFFERSIZE=n größer wählen.

– Rücksetzen dieser Transaktion in einer zwischenzeitlich durchgeführten Deadlockauflösung.

– Eingriff des Datenbankadministrators via DAL (Kommandos ABORT, PERFORM, CLOSE).

– Neue Update-Transaktionen während des Schreibens eines Checkpoints.

– Auftreten eines Datei- oder Programmierfehlers, der durch CANCEL der Transaktion (vorläufig) umgangen werden kann.

– Fehler in einer DML-Anweisung, die nicht für sich rücksetzbar ist und daher den CANCEL der gesamten Transaktion erfordert. Der Datenbankadministrator wurde verständigt (über eine UDS/SQL-Meldung).

– für UDS-D:
Das Rücksetzen der Transaktion kann auch an Fehlern bzw. Administratoreingriffen in einer entfernten Konfiguration (z.B. ABORT, CLOSE CALLS, CLOSE RUN-UNITS, %TERM) oder an Fehlern in der Verbindung zur entfernten Konfiguration liegen.

123 Die Transaktion versucht einen Realm mit READY-USAGE-MODE UPDATE zu eröffnen, der für Änderungen gesperrt ist.
Mögliche Ursachen für diese Sperre sind auf

Konfigurationsebene:

– Die aktuelle Session des independent DBH wurde ohne RLOG-Logging gestartet (PP LOG=NO).

– Das Eröffnen der RLOG-Datei ist misslungen, d.h. das RLOG-Logging ist zur Zeit blockiert.

Datenbankebene:

– Die Datenbank ist als SHARED-RETRIEVAL-Datenbank zugeschaltet.

– Die Datenbank ist keine Original-Datenbank, sondern eine Schattendatenbank.

– Das Eröffnen einer neuen ALOG-Datei ist misslungen, d.h. das AFIM-Logging der Datenbank ist zur Zeit blockiert.

– Die Datenbank wurde vom Datenbankadministrator via DAL-ACCESS-Kommando für Änderungen gesperrt.

– Das DBDIR der Datenbank ist für Änderungen gesperrt (siehe „Realm-Ebene“).

Realm-Ebene:

– Der Realm wurde vom Datenbankadministrator via DAL-ACCESS-Kommando für Änderungen gesperrt.

– Die Transaktion versucht, einen Realm einer entfernten Datenbank zu eröffnen, obwohl die aktuelle Session (wegen PP LOG=NO oder misslungenem Eröffnen der RLOG-Datei) ohne RLOG-Logging fährt.
Dadurch würde die Basis für das Zwei-Phasen-Ende-Protokoll verteilter Transaktionen fehlen.

124 Die Transaktion wurde vom DBH vorzeitig mit CANCEL zurückgesetzt.

Ursache:
Neue Update-Transaktion oder Update-Verarbeitungskette während des Schreibens eines Checkpoints oder während des RLOG-Datei-Wechsels.

Dieser Statuscode wird nur gesetzt, wenn für die aktuelle Session der Ladeparameter PP ORDER-DBSTATUS=SPECIAL angegeben wurde. Andernfalls wird unter den oben genannten Bedingungen der Statuscode 122 gesetzt.

131 Der Database Handler lässt die Transaktion nicht zu, da die beim Laden des Database Handler durch den Ladeparameter TRANSACTION angegebene Anzahl parallel zulässiger Transaktionen bzw. Anwendertasks erschöpft ist.

132 Der Database Handler lässt die Transaktion nicht zu, da die beim Laden des Database Handler durch den Ladeparameter SUBSCHEMA angegebene Anzahl zulässiger Subschemata erschöpft ist.

Statuscodes zur Reihenfolge der DML-Anweisungen

134 Der Database Handler lässt eine DML-Anweisung nicht zu, da keine Transaktion offen ist.

136 Eine DML-Anweisung wird abgewiesen, die zwar zu einer existierenden Transaktion gehört, sich aber an eine Datenbank wendet (eine DB-Referenz liefert) für die aktuell keine Verarbeitungskette der Transaktion existiert.

137 Das Mischen von SQL- und Nicht-SQL-Anweisungen in einer Transaktion ist unzulässig (Ausnahme: Zugriff auf verschiedene UDS/SQL-Konfigurationen über openUTM). Das Mischen von COBOL-DML- und CALL-DML-Anweisungen in einer Verarbeitungskette ist unzulässig.

Statuscodes zum Subschema

141 Die Transaktion hat einen ungültigen bzw. unbekannten Subschema-Namen angegeben oder der Subschema-Name ist in der aktuellen DB-Konfiguration innerhalb der ersten 6 Zeichen nicht eindeutig oder
die betreffende Datenbank ist nicht zugeschaltet.

für UDS-D:
Das angesprochene Subschema ist

- nicht in der lokalen Konfiguration enthalten und nicht in der Verteiltabelle angegeben.
- in der Verteiltabelle angegeben, aber in der entsprechenden UDS/SQL-Konfiguration nicht enthalten.
- in der Verteiltabelle angegeben, aber die entsprechende UDS/SQL-Konfiguration ist nicht erreichbar,
  a) weil der Rechner nicht erreichbar ist
  b) weil die Konfiguration gar nicht oder nicht mit eingeschalteter Verteilung läuft.
- in der Verteiltabelle angegeben, aber gesperrt bzw. die zugehörige Datenbank oder Konfiguration ist gesperrt.

– in der lokalen Konfiguration nicht enthalten und in der lokalen Konfiguration wurde UDS-D nicht gestartet.

Die Anzahl der von dieser Transaktion angesprochenen entfernten Datenbanken überschreitet den Wert PP DISDB.

142 Die Subschema-Beschreibung im DBDIR (SSIA) ist zerstört. BGSSIA-Lauf wiederholen.

144 Die DML-Anweisung spezifiziert ein zur aktuellen READY-Anweisung unterschiedliches Subschema (Subschema-Referenz).

145 Das in der READY-Anweisung angesprochene Subschema kann nicht prozessiert werden, weil es nicht zum aktuellen Stand des Schemas passt (Subschema-DDL-Compilierung und/oder BGSSIA-Lauf nach Datenbankrestrukturierung fehlt)
oder die READY-Anweisung wird abgewiesen, weil die UDS/SQL-Version nicht zur Datenbank passt:

– Die Datenbank wurde für die jahrhundertgerechte Bearbeitung zweistelliger Jahresfelder eingestellt bzw. diese Einstellung wurde nicht vorschriftsmäßig entfernt. Daher darf sie nur mit einer Version ab UDS/SQL V2.0B30 bearbeitet werden.

– Ein Subschema enthält nationale Daten (Unicode: UTF-16, PICTURE N, USAGE NATIONAL). Daher darf es nur mit einer Version ab UDS/SQL V2.5 bearbeitet werden.

146 COBOL-DML: Das Subschema, mit dem das Modul der aktuellen DML-Anweisung übersetzt wurde, entspricht nicht dem aktuellen Stand der Datenbank.
CALL-DML: Das verwendete SSITAB-Modul entspricht nicht dem aktuellen Stand der Datenbank.

Statuscodes zur DBH-Verfügbarkeit

151 Der Database Handler ist noch nicht verfügbar oder wird normal beendet (Beendigung läuft).

152 Der Database Handler wurde abnormal beendet.

154 In UDS/SQL wurde ein nicht behebbarer Fehler erkannt; das Programm sollte beendet werden (STOP RUN bei COBOL-Programmen). Die Transaktion wurde nicht abgeschlossen.

155 Während UDS/SQL eine DML-Anweisung bearbeitet, trifft für die gleiche Transaktion eine weitere DML-Anweisung ein (Entserialisierung).
Mögliche Fehlerursachen:
Asynchrone Aktivitäten des Anwenderprogramms (z.B. DML-Anweisung in STXIT-Routine) oder UDS/SQL-Systemfehler.

Weitere Statuscodes der UDS-Online-Utility

161 Eine Transaktion einer Online-Utility ist bereits auf demselben Realm aktiv.

162 Eine parallel ablaufende User-Transaktion hat eine Online-Realm-Erweiterung angestoßen und damit die Online-Utility temporär behindert.

163 Auf einem temporären Realm ist die Online-Utility nicht erlaubt.

164 Für diesen RELOCATE-Type ist USAGE-MODE EXCLUSIVE UPDATE erforderlich

165 Der angegebene SET ist keine verteilbare Liste

166 Der angegebene Realm ist für die Satzart nicht erlaubt

167 Konkurrierende Änderung einer parallelen User-TA. Die Utility-TA wird zurückgesetzt.

Statuscodes zu FIND/FETCH

183 Der Suchausdruck überschreitet die maximale Länge.

184 Der Temporäre Realm ist nicht vorhanden.

191 Sowohl der Objekt-Set als auch der LIMITED-Set sind dynamisch.

192 Der LIMITED-Set ist leer.

193 FIND/FETCH Format 7:
Der LIMITED-Set enthält eine andere Satzart als der Objekt-Set.

FIND/FETCH Format 4 und 7:
Der Objekt-Set ist dynamisch und enthält eine andere Satzart als die angegebene.

FIND/FETCH Format 3:
Der angegebene Satzname unterscheidet sich von dem im vorangegangenen FIND/FETCH Format 7 angegebenen Satznamen.

194 Vergleichswert oder Sortierfeld hat die Länge 0 oder eine für den Feldtyp nicht erlaubte Länge.

195 Vergleichswert oder Sortierfeld hat unbekannten Feldtyp oder der Vergleichswert enthält nicht typverträgliche Daten.

197 Kein FIND/FETCH Format 7 vorausgegangen.

198 Der CRS des Result-Sets wurde durch eine andere Transaktion aus dem Objekt-Set aus- bzw. in eine andere Occurrence umgehängt.

Statuscodes zur Zusammenarbeit mit openUTM

200 FINISH:
Die FINISH-Anweisung wurde akzeptiert; die Ausführung des FINISH wird jedoch bis zum openUTM-Transaktionsende-Aufruf an die DC-Steuerung (PEND) verzögert. Es werden keine DML-Anweisungen mehr angenommen.

201 Nach dem verzögerten FINISH wurde noch eine weitere DML-Anweisung abgesetzt. Die DML-Anweisung wird ignoriert.

218 Systemübergreifender Deadlock, der sich nur durch Freigeben des openUTM-Anwendertasks auflösen lässt (z.B. durch PEND RS).

Beispiele:

- lokaler UDS/SQL-openUTM-Betrieb:
  Deadlock zwischen UDS/SQL-Betriebsmitteln (Daten) und openUTM-Betriebsmitteln (Tasks).
- Verteilte Verarbeitung über UDS-D oder openUTM-D:
  Deadlock zwischen UDS/SQL-Betriebsmitteln (Daten) und/oder openUTM-Betriebsmitteln (Tasks).

Die Erkennung solcher Deadlocks erfolgt über eine Zeitüberwachung von Wartesituationen (PP DEADTIME). Nach Ablauf dieses Zeitlimits wird der Statuscode 218 angezeigt, auch wenn u.U. kein wirklicher Deadlock vorliegt.

Statuscodes zur LOOK-Funktion

781 Element nicht gefunden oder unbekannter Realmname bei der Online-Utility.

782 Es existiert kein nächstes Element.

783 Ein Element der Liste nicht gefunden.

784 Die eingegebene Feldreferenz existiert nicht. Es wurde die Beschreibung mit der nächstniedrigeren Feldreferenz ausgegeben.

785 Der Ergebnisvektor einer zusammengesetzten LOOKC-Funktion muss durch eine lückenlose Folge von entsprechenden LOOKC-Anweisungen abgerufen werden.

786 Satzart mit diesem Subschema nicht prozessierbar, da sie Daten eines Typs enthält, der dem Anwenderprogramm nicht bekannt ist.

789 Das angegebene Subschema existiert nicht.

Statuscodes zur Zuordnung von Speicherplatz oder Database Key

802 Der Speicherplatz im Realm ist erschöpft oder eine aktivierte Online-Realm-Erweiterung ist gescheitert. Der betroffene Satz kann nicht gespeichert oder in eine Set-Occurrence eingefügt werden.

804 Zur Speicherung eines neuen Satzes ist kein Database Key mehr verfügbar, oder eine aktivierte Online-Realm-Erweiterung ist gescheitert.

805 Der System-Adressraum des DBH ist erschöpft. Das Tabellenwerk des DBH kann nicht mehr dynamisch erweitert werden. Der Datenbankadministrator wurde verständigt.

Statuscodes zu variablen Feldern und Komprimierung

888 Die Länge des variablen Feldes ist größer als im Schema definiert oder negativ.

898 STORE/MODIFY Format-2 ist bei variablen Feldern nicht erlaubt.

899 STORE:
Die Anzahl der Felder, die gespeichert werden sollen, ist so groß, dass die Größe des komprimierten Satzes größer ist als eine Seite.

GET:
Eines der gewünschten Felder ist im komprimierten Satz in der Datenbank nicht vorhanden.

MODIFY Format-1:
Dieses Format ist nicht erlaubt, falls der angesprochene Satz in komprimierter Form vorliegt.

MODIFY Format-2:
Eines der Felder, das verändert werden soll, ist im komprimierten Satz nicht vorhanden.

Statuscodes bezüglich Zugriffsrechten

901 Zugriff auf einen Realm, Record oder Set innerhalb der Benutzergruppe nicht erlaubt, oder die Dienstprogramme ONLINE-PRIVACY bzw. ONLINE-UTILITY versuchen auf eine Datenbank zuzugreifen, die nicht in der Ablaufkennung des Dienstprogramms steht. Es ist nicht möglich, mittels Setzen des P-Parameters PRIVACY-CHECK auf OFF dieses Verhalten der Dienstprogramme zu umgehen.

950 Benutzergruppe unbekannt (siehe Handbuch „Aufbauen und Umstrukturieren“, BPRIVACY).

954 Für die Benutzergruppe sind keine Zugriffsberechtigungen definiert.

Statuscodes der CALL-DML

DML-Wahleintrag-Fehler:

C00 Der angegebene Funktionsname ist nicht korrekt.

C01 Die angegebene Funktionswahl ist bei dem angegebenen Funktionsnamen nicht erlaubt.

C02 Die angegebene Zusatzwahl ist bei der angegebenen Kombination von Funktionsname und Funktionswahl nicht erlaubt oder sie ist syntaktisch fehlerhaft.

Satzname-Fehler:

C03 Der angegebene Satzname ist im betreffenden Subschema nicht vorhanden oder nicht eindeutig.

C04 Ein obligatorischer Satzname wurde nicht angegeben.

Setname-Fehler:

C05 Der angegebene Setname ist im aktuellen Subschema nicht vorhanden oder nicht eindeutig.

C06 Syntaxfehler in der Setnamenleiste
(zu viele Setnamen; Trennung oder Abschluss der Setnamen fehlerhaft; Setname tritt mehrfach auf)

Realm-Namen-Fehler:

C07 Der angegebene Realm-Name ist im aktuellen Subschema nicht vorhanden oder nicht eindeutig.

C08 Syntaxfehler in der Realm-Namenleiste
(zu viele Realm-Namen; Trennung oder Abschluss der Realm-Namen fehlerhaft; Realm-Name tritt mehrfach auf)

Feldnamen-Fehler:

C09 Der angegebene Feldname ist im betreffenden Satz des aktuellen Subschemas nicht vorhanden oder nicht eindeutig.

C10 Syntaxfehler in der Feldnamenleiste
(zu viele Feldnamen; Trennung oder Abschluss der Feldnamen fehlerhaft)

IF-Ergebnis:

C11 Die IF-Bedingung trifft nicht zu.
C11 ist nicht als Fehlercode, sondern als Ergebnis der IF-DML-Anweisung anzusehen; 000, wenn Bedingung zutrifft

Suchausdruck-Fehler:

C20 Der Suchausdruck enthält zu viele Suchbedingungen.

C21 Eine NXT-Suchbedingung nach einem OR-Operator ist verboten.

C22 Das Trennzeichen vor und hinter dem Feldnamen bzw. Vergleichsoperator jeder Suchbedingung muss jeweils ein Zwischenraum sein.

C23 Die Anzahl der Klammern einer NXT-Suchbedingung muss gleich Null sein.

C24 Die Maske einer Suchbedingung darf nur aus den Zeichen 0 und 1 bestehen und muss mit einem Zwischenraum abgeschlossen sein.

C25 Eine NXT-Suchbedingung darf nicht innerhalb von Klammern stehen.

C26 Die Länge der Maske einer Suchbedingung muss gleich der Länge des Feldes sein.

C27 NXT-Suchbedingungen dürfen nur am Ende eines Suchausdrucks stehen.

C28 Eine Suchbedingung ist nicht mit _OR_, _AN_ oder _END abgeschlossen.

C29 Die Länge des Wertes in einer Suchbedingung ist inkorrekt.

C30 Die Anzahl der rechten Klammern einer Suchbedingung ist nicht numerisch.

C32 In einem Suchausdruck sind mehr linke als rechte Klammern vorhanden.

C33 In einer NXT-Suchbedingung ist die NEQ-Beziehung verboten.

C34 Der Beziehungs-Operator in einer Suchbedingung ist nicht korrekt.

C35 Die Anzahl der linken Klammern einer Suchbedingung ist nicht numerisch.

C37 In einer Suchbedingung sind zu viele rechte Klammern angegeben.

C38 Der Beziehungs-Operator in einer Suchbedingung wird nicht von einem Zwischenraum gefolgt.

C39 Der Feldname einer Suchbedingung ist im aktuellen Subschema nicht vorhanden oder nicht eindeutig.

C40 Der Feldtyp einer Suchbedingung ist abdruckbar numerisch, der zugehörige Vergleichswert nicht.

C41 Der Feldtyp einer Suchbedingung ist dezimal gepackt, der zugehörige Vergleichswert nicht.

C42 Suchbedingungen sind für diesen Feldtyp nicht erlaubt.

Retaining-Wahleintrag-Fehler:

C61 Der angegebene Retaining-Wahleintrag (Spezialparameter-1) ist nicht korrekt.

C62 Ein angegebener Retaining-Setname (Spezialparameter-1) ist im aktuellen Subschema nicht vorhanden oder nicht eindeutig.

C63 Syntaxfehler in der Retaining-Setnamenleiste
(zu viele Setnamen; Trennung oder Abschluss der Setnamen fehlerhaft; Setname tritt mehrfach auf)

Andere Fehler:

C66 Das SSITAB-Modul des Subschemas ist nicht identifizierbar oder der angegebene Subschemaname stimmt nur in den ersten 6 Zeichen mit dem im SSITAB-Modul vermerkten Subschemanamen überein, nicht aber in voller Länge.
BCALLSI-Lauf durchführen.

C72 Die Ganzzahl der Positionsangabe eines FIND4/FTCH4-Aufrufs darf nicht Null sein.

Spezielle FIND7A/FTCH7A-Fehler:

C74 Der angegebene Name des begrenzenden Sets ist im aktuellen Subschema nicht vorhanden oder nicht eindeutig.

C75 Der angegebene Name des Ergebnis-Sets ist im aktuellen Subschema nicht vorhanden oder nicht eindeutig.

Spezielle LOOKC-Fehler:

C80 Die Anzahl der LOOKC-Blöcke muss zwischen 1 und 255 (einschließlich) liegen.

Benutzer-Kommunikations-Fehler:

C90 Der vom Konverter-Modul UDSCDML benötigte Arbeitspuffer kann nicht im notwendigen Umfang zur Verfügung gestellt werden. Gegebenenfalls muss der Communication Pool vergrößert werden (siehe Handbuch „Datenbankbetrieb“).

C91 Der Fehlerausgang DSCEXT wurde nicht definiert.

C94 Das Konverter-Modul UDSCDML ist nicht vorhanden.

C95 Das von BCALLSI erstellte SSITAB-Modul ist nicht vorhanden, oder konnte nicht in den Speicher geladen werden (z.B. wegen Speicherplatzmangels).

C98 Es wird versucht, ACCPTL, FIND1L, FTCH1L, STORE1L oder STORE2L mit einem SSITAB-Modul auszuführen, das vor UDS/SQL V2.0 erzeugt wurde, oder mit einem Subschema "FORM IS OLD". Für die Ausführung der genannten Funktionen wird ein SSITAB-Modul einer Version ab UDS/SQL V2.0 benötigt.

C99 Das SSITAB-Modul ist ungültig oder passt nicht zur Version des CALL-DML-Umsetzers.

Zulässigkeitsprüfung für DML-Anweisungen auf Grund der Subschemastruktur:

P01 Ein FIND2/FTCH2 mit Wahleintrag ANY... ist nur erlaubt, wenn LOCATION MODE IS CALC spezifiziert ist und alle Keys der Satzart im Subschema vorhanden sind.

P02 Ein FIND2/FTCH2 mit Wahleintrag DUPLIC ist nur erlaubt, wenn LOCATION MODE IS CALC und DUPLICATES ARE ALLOWED spezifiziert sind und alle Keys der Satzart im Subschema vorhanden sind.

P03 Bei dem aktuellen FIND3/FTCH3 sind Duplikate nicht erlaubt.

P04 Ein FIND7A/FTCH7A ist nur erlaubt, wenn die angesprochene Satzart Member im angegebenen Set ist.

P05 Ein FIND7A/FTCH7A bei SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER ist

- im Zusammenhang mit LOCATION MODE IS DIRECT nur erlaubt, wenn das betroffene Feld im Subschema vorhanden ist.
- im Zusammenhang mit LOCATION MODE IS CALC nur erlaubt, wenn alle Keys der Satzart im Subschema vorhanden sind.

P06 Ein FIND4/FTCH4 bzw. FIND5/FTCH5 ist nur erlaubt, wenn die angegebene Satzart Member des angegebenen Sets ist.

P07 Ein FIND4/FTCH4 bzw. FIND5/FTCH5 ist nur erlaubt, wenn die angegebene Satzart im angegebenen Realm zulässig ist.

P08 Ein FIND6/FTCH6 ist nur erlaubt, wenn es sich nicht um einen SYSTEM Set handelt.

P09 Die spezifizierte Speicherungsform des Sets lässt kein CONNEC bzw. DISCON zu oder bei einem DISCON ALLFRM ist der angegebene Set kein Dynamic Set.

P10 In der Setnamenleiste eines CONNEC bzw. DISCON sind nur Sets zulässig, die untereinander dieselbe Satzart als Member haben.

P11 Der Current of Rununit muss bei einem CONNEC bzw. DISCON der Membersatzart des angegebenen Sets angehören.

P12 Bei einem CONNEC TO-ALL muss mindestens ein Set mit der angesprochenen Satzart im Subschema enthalten sein, die nicht MANDATORY AUTOMATIC ist. Bei einem DISCON FRMALL muss die angesprochene Satzart OPTIONAL Member in mindestens einem Set des Subschemas sein.

P13 Der angegebene MODIF1/2 ist nicht erlaubt.

P14 Der angegebene STORE1/2 ist nicht erlaubt.

P15 Der angegebene ERASEC ist nicht erlaubt.

P16 Der in der RESULT- und/oder LIMITED-Klausel angegebene Set ist kein dynamischer Set.

10.2 Schemabeschreibung von ARTIKELVERSAND der Beispieldatenbank VERSAND

```
SCHEMA NAME IS                          ARTIKELVERSAND
    PRIVACY LOCK FOR COPY IS            "VERSANDKEY".
*
*
*
 AREA NAME IS                            AUFTRAGSRLM.
 AREA NAME IS                            BESTELLRLM.
 AREA NAME IS                            KLEIDUNG.
 AREA NAME IS                            HAUSHALT.
 AREA NAME IS                            SPORT.
 AREA NAME IS                            LEBENSMITTEL.
 AREA NAME IS                            SPIELE-HOBBY.
 AREA NAME IS                            SCHREIBWAREN.
 AREA NAME IS                            ARTIKELRLM.
 AREA NAME IS                            SUCHRLM
                                         AREA IS TEMPORARY.
*
*
*
 RECORD NAME IS                          KUNDE
     LOCATION MODE IS DIRECT-LONG KUNDEN-NR OF KUNDE
     WITHIN AUFTRAGSRLM.
*
 01  KUNDEN-NAME                        TYPE IS CHARACTER 30.
 01  KUNDEN-VORNAME                     TYPE IS CHARACTER 30.
 01  KUNDEN-NR                          TYPE IS DATABASE-KEY-LONG.
*
*
 RECORD NAME IS                          AUFTRAG
     WITHIN AUFTRAGSRLM.
*
 01  AUFTR-NR                           PICTURE IS 9(4).
 01  AUFTR-JAHR                         PICTURE IS 99.
 01  AUFTR-MONAT                        PICTURE IS 99.
 01  AUFTR-TAG                          PICTURE IS 99.
 01  AUFTR-STATUS                       PICTURE IS X.
*
*
 RECORD NAME IS                          AUFTR-POS
     WITHIN AUFTRAGSRLM.
```

```
*
 01  AUFTR-POS-NR                    PICTURE IS 99.
 01  AUFTR-MENGE                     TYPE IS DECIMAL 6.
 01  KENNZ-RATENZAHLUNG              PICTURE IS X.
 01  AUFTR-POS-STATUS                PICTURE IS X.
*
*
 RECORD NAME IS                       RATENZAHLUNG
     WITHIN AUFTRAGSRLM
     SEARCH KEY IS NEXT-RATE-JAHR, NEXT-RATE-MONAT, NEXT-RATE-TAG
        USING INDEX NAME IS SEARCH-TAB-RATENZAHLUNG
        DUPLICATES ARE ALLOWED.
*
 01  AUFTR-NR                        PICTURE IS 9(4).
 01  AUFTR-POS-NR                    PICTURE IS 99.
 01  GESAMTPREIS-RATE                TYPE IS DECIMAL 9,2.
 01  EINZELRATE                      TYPE IS DECIMAL 7,2.
 01  RESTBETRAG                      TYPE IS DECIMAL 9,2.
 01  NEXT-RATE-JAHR                  PICTURE IS 99.
 01  NEXT-RATE-MONAT                 PICTURE IS 99.
 01  NEXT-RATE-TAG                   PICTURE IS 99.
*
*
 RECORD NAME IS                       ARTIKELART
     WITHIN KLEIDUNG, HAUSHALT, SPORT, LEBENSMITTEL, SPIELE-HOBBY,
        SCHREIBWAREN AREA-ID IS RLMAUSWAHL-1
     SEARCH KEY IS ART-BEZ USING CALC
        NAME IS SEARCH-TAB-ARTIKELART DUPLICATES ARE ALLOWED.
*
 01  ART-BEZ                         TYPE IS CHARACTER 25.
*
*
 RECORD NAME IS                       ARTIKELAUSWAHL
     WITHIN KLEIDUNG, HAUSHALT, SPORT, LEBENSMITTEL, SPIELE-HOBBY,
        SCHREIBWAREN AREA-ID IS RLMAUSWAHL-2
     SEARCH KEY IS WAHL-KRIT USING INDEX
        NAME IS SEARCH-TAB-ARTIKELAUSWAHL DUPLICATES ARE ALLOWED.
*
 01  WAHL-KRIT                       TYPE IS CHARACTER 25.
*
*
 RECORD NAME IS                       ARTIKELBESCHR
     LOCATION MODE IS CALC USING BEZEICHNUNG
        DUPLICATES ARE ALLOWED
```

```
      WITHIN KLEIDUNG, HAUSHALT, SPORT, LEBENSMITTEL, SPIELE-HOBBY,
         SCHREIBWAREN AREA-ID IS RLMAUSWAHL-3.
*
 01  ART-NR                          PICTURE IS 9(6).
 01  BEZEICHNUNG                     TYPE IS CHARACTER 40.
 01  MATERIAL                        OCCURS 4 TIMES.
     02 PROZENT                      PICTURE IS 99.
     02 MAT-ABK                      PICTURE IS X.
 01  LAENGENFELD                     TYPE IS BINARY 15.
 01  ARTIKELERLAEUTERUNG             PICTURE IS LX(500)
                                     DEPENDING ON LAENGENFELD.
*
*
 RECORD NAME IS                      ARTIKEL
     LOCATION MODE IS CALC USING ART-NR, FARB-NR, GROESSE
         DUPLICATES ARE NOT ALLOWED
      WITHIN KLEIDUNG, HAUSHALT, SPORT, LEBENSMITTEL, SPIELE-HOBBY,
         SCHREIBWAREN AREA-ID IS RLMAUSWAHL-4
      SEARCH KEY IS ART-NR-LIEFER, FARB-NR-LIEFER, GROESSE
         USING CALC NAME IS SEARCH-TAB-ARTIKEL-1
         DUPLICATES ARE NOT ALLOWED
      SEARCH KEY IS BEZEICHNUNG USING CALC
         NAME IS SEARCH-TAB-ARTIKEL-2 DUPLICATES ARE ALLOWED.
 *
  01  ART-NR                         PICTURE IS 9(6).
  01  FARB-NR                        PICTURE IS 99.
  01  BEZEICHNUNG                    TYPE IS CHARACTER 40.
  01  ART-NR-LIEFER                  PICTURE IS 9(4).
  01  FARB-NR-LIEFER                 PICTURE IS 99.
  01  GROESSE                        PICTURE IS 99.
  01  PREIS                          TYPE IS DECIMAL 7,2.
  01  PREIS-RATENZAHLUNG             TYPE IS DECIMAL 7,2.
  01  MAX-BESTAND                    TYPE IS DECIMAL 10.
  01  MIN-BESTAND                    TYPE IS DECIMAL 3.
  01  AKT-BESTAND                    TYPE IS DECIMAL 10.
  01  STATISTIK                      TYPE IS DECIMAL 15.
  01  KENNZ-NICHT-LIEFERBAR          PICTURE IS X.
 *
 *
  RECORD NAME IS                     TEILMENGE
      WITHIN HAUSHALT, SPORT AREA-ID IS RLMAUSWAHL-5.
 *
  01  MENGE                          PICTURE IS 99.
 *
```

```
*
 RECORD NAME IS                          FARBEN
     WITHIN ARTIKELRLM
     SEARCH KEY IS FARB-BEZ USING CALC DUPLICATES ARE NOT ALLOWED
     SEARCH KEY IS FARB-NR USING CALC DUPLICATES ARE NOT ALLOWED.
*
 01  FARB-NR                            PICTURE IS 99.
 01  FARB-BEZ                           TYPE IS CHARACTER 20.
*
*
 RECORD NAME IS                          MATERIALIEN
     WITHIN ARTIKELRLM
     SEARCH KEY IS MAT-ABK USING INDEX
        NAME IS SEARCH-TAB-MATERIAL-1 DUPLICATES ARE NOT ALLOWED
     SEARCH KEY IS MAT-BEZ USING INDEX
        NAME IS SEARCH-TAB-MATERIAL-2 DUPLICATES ARE NOT ALLOWED.
*
 01  MAT-ABK                            TYPE IS CHARACTER 1.
 01  MAT-BEZ                            TYPE IS CHARACTER 20.
*
*
 RECORD NAME IS                          LIEFERANT
     LOCATION MODE IS CALC USING LIEFER-NR, LIEFER-NAME
        DUPLICATES ARE NOT ALLOWED
     WITHIN BESTELLRLM.
*
 01  LIEFER-NR                          PICTURE IS 9(5).
 01  LIEFER-NAME                        TYPE IS CHARACTER 30.
 01  LIEFER-PLZ                         TYPE IS CHARACTER 4.
 01  LIEFER-STADT                       TYPE IS CHARACTER 30.
 01  LIEFER-STRASSE                     TYPE IS CHARACTER 30.
 01  LIEFER-HAUSNR                      TYPE IS CHARACTER 3.
 01  LIEFER-TEL                         PICTURE IS 9(12).
 01  LIEFER-POSTFACH                    PIC 9(4).
 01  LIEFER-FERNSCHR                    PIC 9(12).
*
*
 RECORD NAME IS                          BESTELLUNG
     WITHIN BESTELLRLM.
*
 01  BEST-NR                            PICTURE IS 9(4).
 01  BEST-JAHR                          PICTURE IS 99.
 01  BEST-MONAT                         PICTURE IS 99.
 01  BEST-TAG                           PICTURE IS 99.
```

```
*
*
 RECORD NAME IS                          BESTELL-POS
     WITHIN BESTELLRLM.
*
 01  BEST-POS-NR                        PICTURE IS 99.
 01  BEST-MENGE                         TYPE IS DECIMAL 10.
*
*
*
 SET NAME IS                             ERTEILTE-AUFTRAEGE
     ORDER IS SORTED INDEXED BY DEFINED KEYS
        DUPLICATES ARE NOT ALLOWED
     OWNER IS KUNDE.
 MEMBER IS AUFTRAG OPTIONAL AUTOMATIC
     ASCENDING KEY IS AUFTR-NR
     SEARCH KEY IS AUFTR-JAHR, AUFTR-MONAT, AUFTR-TAG USING INDEX
        NAME IS SEARCH-TAB-ERT-AUFTR DUPLICATES ARE ALLOWED
     SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
 SET NAME IS                             AUFTR-INHALT
     ORDER IS SORTED INDEXED BY DEFINED KEYS
        DUPLICATES ARE NOT ALLOWED
     OWNER IS AUFTRAG.
 MEMBER IS AUFTR-POS MANDATORY AUTOMATIC
     ASCENDING KEY IS AUFTR-POS-NR
     SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
 SET NAME IS                             NOCH-ZU-BEZAHLEN
     ORDER IS LAST
     OWNER IS KUNDE.
 MEMBER IS AUFTR-POS OPTIONAL AUTOMATIC
     SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
 SET NAME IS                             RATENKAUF
     ORDER IS LAST
     OWNER IS KUNDE.
 MEMBER IS RATENZAHLUNG MANDATORY AUTOMATIC
     SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
```

```
 SET NAME IS                          ANGEBOT
     ORDER IS SORTED INDEXED BY DEFINED KEYS
        DUPLICATES ARE ALLOWED
     OWNER IS ARTIKELART.
 MEMBER IS ARTIKELBESCHR MANDATORY AUTOMATIC
     ASCENDING KEY IS BEZEICHNUNG
     SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
 SET NAME IS                          NAEHERE-AUSWAHL
     ORDER IS SORTED INDEXED BY DEFINED KEYS
        DUPLICATES ARE ALLOWED
     OWNER IS ARTIKELAUSWAHL.
 MEMBER IS ARTIKELBESCHR MANDATORY AUTOMATIC
     ASCENDING KEY IS BEZEICHNUNG
     SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
 SET NAME IS                          BESTELLANGABEN
     ORDER IS SORTED INDEXED BY DEFINED KEYS
        DUPLICATES ARE NOT ALLOWED
     OWNER IS ARTIKELBESCHR.
 MEMBER IS ARTIKEL MANDATORY AUTOMATIC
     ASCENDING KEY IS FARB-NR, GROESSE
     SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
 SET NAME IS                          MIN-BESTAND-ERREICHT
     ORDER IS SORTED INDEXED BY DEFINED KEYS
        DUPLICATES ARE NOT ALLOWED
     OWNER IS SYSTEM.
 MEMBER IS ARTIKEL OPTIONAL MANUAL
     ASCENDING KEY IS ART-NR, FARB-NR, GROESSE.
*
*
 SET NAME IS                          ENTHAELT
     ORDER IS NEXT
     OWNER IS ARTIKEL.
 MEMBER IS TEILMENGE MANDATORY AUTOMATIC
     SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
 SET NAME IS                          ENTHALTEN-IN
     ORDER IS NEXT
```

```
     OWNER IS ARTIKEL.
 MEMBER IS TEILMENGE MANDATORY AUTOMATIC
     SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER
        ALIAS FOR   ART-NR     IS   ERSATZ-ART-NR
        ALIAS FOR   FARB-NR    IS   ERSATZ-FARB-NR
        ALIAS FOR   GROESSE    IS   ERSATZ-GROESSE.
*
*
 SET NAME IS                          LIEFERANTEN
     ORDER IS SORTED INDEXED BY DEFINED KEYS
        DUPLICATES ARE NOT ALLOWED
     OWNER IS SYSTEM.
 MEMBER IS LIEFERANT MANDATORY AUTOMATIC
     ASCENDING KEY IS LIEFER-NAME, LIEFER-NR.
*
*
 SET NAME IS                          LIEFERBARE-ARTIKEL
     ORDER IS SORTED INDEXED BY DEFINED KEYS
        DUPLICATES ARE ALLOWED
     OWNER IS LIEFERANT.
 MEMBER IS ARTIKEL MANDATORY AUTOMATIC
     ASCENDING KEY IS BEZEICHNUNG
     SEARCH KEY IS KENNZ-NICHT-LIEFERBAR USING INDEX
        NAME IS SEARCH-TAB-LIEF-ART DUPLICATES ARE ALLOWED
     SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
 SET NAME IS                          BESTELLTE-ARTIKEL
     ORDER IS LAST
     OWNER IS ARTIKEL.
 MEMBER IS AUFTR-POS MANDATORY AUTOMATIC
     SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
 SET NAME IS                          NACHBESTELLTE-ARTIKEL
     ORDER IS LAST
     OWNER IS ARTIKEL.
 MEMBER IS BESTELL-POS MANDATORY AUTOMATIC
     SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
 SET NAME IS                          ABGEGEBENE-BEST
     ORDER IS LAST
     OWNER IS LIEFERANT.
```

```
 MEMBER IS BESTELLUNG MANDATORY AUTOMATIC
     SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
 SET NAME IS                          EMPFANGENE-BEST
     ORDER IS FIRST
     OWNER IS LIEFERANT.
 MEMBER IS BESTELLUNG MANDATORY MANUAL
     SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
 SET NAME IS                          BESTELL-INHALT
     ORDER IS NEXT
     OWNER IS BESTELLUNG.
 MEMBER IS BESTELL-POS MANDATORY AUTOMATIC
     SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
*
 SET NAME IS                          ERGEBNISSET
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
*
 SET NAME IS                          LIMITED-SET
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
*
*
 SET NAME IS                          IQL-DYN1
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
*
 SET NAME IS                          IQL-DYN2
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
*
 SET NAME IS                          IQL-DYN3
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
```

```
*
 SET NAME IS                          IQL-DYN4
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
*
 SET NAME IS                          IQL-DYN5
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
*
 SET NAME IS                          IQL-DYN6
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
*
 SET NAME IS                          IQL-DYN7
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
*
 SET NAME IS                          IQL-DYN8
     SET IS DYNAMIC
     ORDER IS IMMATERIAL
     OWNER IS SYSTEM.
*
*
*
```

10.3 Returncodes bei UDS/SQL-openUTM

In der DB Trace Information sind in 4 Bytes taskspezifisch DB-Returncodes enthalten (siehe openUTM-Handbuch „Meldungen, Test und Diagnose (BS2000)“, DB-DIAGAREA). Diese Codes sind versionsabhängig und gelten dann auch nur für diese Version.

Die Returncodes sind aufgebaut wie folgt: **abcd**

a enthält:

X’04’ Fehler bei Eventing (ENABLE)

X’08’ Serialisierungsfehler

X’0C’ Fehler bei Eventing

X’10’ Fehler bei LINK

X’14’ Communication Pool enthält nicht mehr genügend freien Bereich

X’18’ interner Fehler im UDSCON

X’1C’ Memory-Fehler

X’20’ unbekanntes Schlüsselwort

X’24’ START-Parameter nicht für UDS/SQL

X’28’ Längenangabe bei START-Parametern zu klein

X’2C’ DVS-Fehler bei Status-Abfrage

X’30’ kein Applikations-ENTRY verfügbar

X’34’ angesprochener Applikations-ENTRY fehlt

X’38’ Communication Pool wurde normal geschlossen

X’3C’ Communication Pool wurde abnormal geschlossen

X’40’ unbehebbarer Fehler in Memory-Verwaltung

X’44’ Fehler in der Status-Meldung

X’48’ Status der Transaktion nicht eindeutig

X’4C’ Fehler bei FORWARD Eventing

X’50’ Verletzung von Sicherheitsanforderungen

X’54’ Fehler im Programm-Management

X’58’ Fehler im Ergebnistransfer

X‘5C‘ Task-Deadlock in TIAM-Anwendung

X‘60‘ Fehler in Transferdaten

X‘64‘ Lifetime-Überschreitung bei Auftrag an Mastertask

b enthält:

Zusatzinformation für die Diagnose (bei einigen Fehlern)

c enthält:

Zusatzinformation für die Diagnose (bei einigen Fehlern)

In einigen Fehlersituationen, die durch d schon hinreichen eingegrenzt sind, wird zu Diagnosezwecken eine Zusatzinformation in abc angegeben.

d enthält:

X’00’ Funktion erfolgreich durchgeführt

X’04’ falscher Operationscode

X’08’ interner Fehler (siehe Bedeutung bei a)

X’0C’ keine Verbindung zu UDS/SQL-Subtasks

X’10’ UDS/SQL beendet

X’14’ Transaktion existiert nicht (mehr)

X’18’ Transaktion abnormal beeendet

X’1C’ Eröffnung einer Transaktion scheitert mangels Tabellen

X’20’ Transaktionsbeendigung scheitert

X’24’ Fehler bei START-Parameter

X’28’ falsche Beendigung einer PTC-Transaktion

X’2C’ fehlerhafte Bearbeitung

X’30’ 2. PTC vom Anwender

X’34’ Fehler durch Anwender bei CALL-DML

X’38’ fehlerhafte Bearbeitung beim FINISH

X’3C’ fehlerhafte Bearbeitung beim FINISH WITH CANCEL

X’40’ Fehler bei Status-Anfrage

X’44’ schwerwiegender CALL-DML-Fehler

X’48’ READY-Fehler bei KDBS

X’4C’ keine Verbindung möglich - alle Kanäle belegt

X’50’ falscher Kommunikationsname

X’54’ angesprochener Communication Pool existiert nicht

X’58’ Communication Pool noch nicht fertig eingerichtet

X’5C’ Task existiert nicht für den ein asynchroner CANCEL versucht wurde

X’60’ Kanal aktiv

X’64’ Start BIB: Anwender wünscht Verbindung

X’68’ Stop BIB: Anwender wünscht Trennen der Verbindung

X’6C’ Anwender ist bereits mit UDS/SQL verbunden

X’70’ Versionsmix der UDS/SQL-Module

X’74’ Fehler bei verteilter Verarbeitung

X’78’ unzulässige BS2000-Version

X’7C’ Subschema-Wechsel während der Transaktion

X’80’ Serialisierungsfehler

X’84’ die eigene Task kann nicht asynchron abgebrochen werden

X’88’ für CONTINUE TA lag kein BREAK TA vor

X’8C’ falsche Parameter von Anwender

X’90’ Deadlock

X’94’ READY für Transaktion fehlt

X’98’ schwerer Fehler in offener PTT

X’9C’ Fehler bei Mastertask-Meldung

X’A0’ Fehler bei Update-Erkennung

X’A4’ PETA nur für Update-Transaktion möglich

X’A8’ Eröffnung eines SQL-Vorgangs scheitert mangels Tabellen

X’AC’ es konnte kein SQL-OUTPUT erzeugt werden

X’B0’ im geladenen DBH ist SQL nicht verfügbar

X’B4’ die Länge im Vorgangsmemory ist unkorrekt verändert

X’B8’ UDS/SQL ist nicht im AMODE3 ansprechbar

X’BC’ Anwendung und AMODE passen nicht zusammen

X’C0’ Aufruf nicht erlaubt in sicherer UDS/SQL-Konfiguration

X’C4’ Unzulässige Modulbibliothek

X’C8’ Kein Anschluss an UDS/SQL-Konfiguration möglich

X’CC’ Inkonsistenter DML-Auftrag

10.4 Zusätzliche Diagnoseinformation bei openUTM

openUTM dokumentiert eingetretene Ereignisse in taskspezifischen Trace-Bereichen, die zyklisch beschrieben werden. Dabei werden auch Aufträge an das Datenbanksystem dokumentiert. Für UDS/SQL ist inbesondere das Feld 'Secondary DB Trace Information' relevant. Dort legt UDS/SQL Daten über den einzelnen Auftrag ab, die Sie zur Analyse von Abläufen und zur Diagnose von Fehlersituationen nutzen können. Diese Informationen sind aber teilweise nur mit anderen Diagnoseunterlagen (z. B. Dump) nutzbar, da die zugrunde liegenden Datenfelder nur UDS/SQL-intern genutzt werden. Insofern ist die Interpretation der Felder nur in Verbindung mit der zugrunde liegenden Version von UDS/SQL möglich.

Das Feld 'Secondary DB Trace Information' ist 32 byte lang. Es ist Bestandteil eines Trace-Satzes (DB-Record) der sogenannten DB-DIAGAREA. Bei der openUTM Version V5.3 ist der DB-Record in der UTM-DIAGAREA enthalten
(siehe openUTM-Handbuch „Meldungen, Test und Diagnose (BS2000)“, DB-DIAGAREA bzw. UTM-DIAGAREA).

Die ’Secondary DB Trace Information’ von UDS/SQL ist wie folgt aufgebaut:

Byte	**Bedeutung**	
1-4	Version Der Versionsstring (’U01␣’, ’U02␣’ oder ’U03␣’) dient zur Identifizierung der Trace Information und zur Unterscheidung bei logisch gleichen, aber im Format unterschiedlichen Informationen.	
5-6	Art des Auftrags Mit zwei Zeichen wird logisch die Art des Auftrags von openUTM an UDS/SQL dokumentiert.	
	CB	COBOL-DML
	CD	CALL-DML (inkl. KDBS-Auftrag)
	CN	Konnektierung
	DC	Diskonnektierung
	FN	Transaktionsbeendigung
	PA	Übergabe von Startparametern
	PB	Besonderer Auftrag des COBOL Laufzeitsystems
	RB	Taskfortsetzung einer offenen TA
	SB	Taskunterbrechung bei offener TA
	SQ	SQL-Auftrag
	ST	Status-Anfrage von openUTM
7	openUTM-Opcode 1	
8	openUTM-Opcode 2	
9-32	Unterschiedliche Bedeutungen in Abhängigkeit der in Byte 1-6 definierten Version und Art des Auftrags, siehe folgende Tabellen.	

Tabelle 41: Aufbau des Feldes ’Secondary DB Trace Information’

In den folgenden Tabellen werden die Bedeutungen der Bytes 9-32 des Feldes ’Secondary DB Trace Information’ für die unterschiedlichen Versionen und Auftragsarten aufgeführt.

Byte 1-6: U01 CB

Byte	Bedeutung
9-12	dynamisch vergebene UDS/SQL-interne Transaktions-ID
13	DML-Auftragskennzeichen 1 im BIB
14	DML-Auftragskennzeichen 2 im BIB
15	dynamisch vergebene UDS/SQL-interne Nummer der im DML angesprochenen Satzart bei BIBs alter Art
16	dynamisch vergebene UDS/SQL-interne Nummer von im DML genutztem Set bzw. Realm bei BIBs alter Art
17-19	Status-Code des abgewickelten DML
20	Kennzeichen, ob der in Byte 17-19 ausgewiesene Status-Code mit dem in der BIB an den Anwender übergebenen übereinstimmt (’O’) oder nicht (’B’)
21	dynamisch vergebene UDS/SQL-interne Datenbank-Id
22	dynamisch vergebene UDS/SQL-interne Datenbank-Id in der Remote-Konfiguration
23-26	dynamisch vergebene UDS/SQL-interne Subschema-Referenz
27-32	Subschema-Name

Byte 1-6: U02 CB

Byte	Bedeutung
9-12	dynamisch vergebene UDS/SQL-interne Transaktions-ID
13	DML-Auftragskennzeichen 1 im BIB
14	DML-Auftragskennzeichen 2 im BIB
15-16	dynamisch vergebene UDS/SQL-interne Nummer der im DML angesprochenen Satzart bei BIBs neuer Art
17-19	Status-Code des abgewickelten DML
20	Kennzeichen, ob der in Byte 17-19 ausgewiesene Status-Code mit dem in der BIB an den Anwender übergebenen übereinstimmt (’O’) oder nicht (’B’)
21	dynamisch vergebene UDS/SQL-interne Datenbank-ID
22	dynamisch vergebene UDS/SQL-interne Datenbank-ID in der Remote-Konfiguration
23-26	dynamisch vergebene UDS/SQL-interne Subschema-Referenz
27-32	Subschema-Name

Byte 1-6: U01 CD

Byte	Bedeutung
9-12	dynamisch vergebene UDS/SQL-interne Transaktions-ID
13	DML-Auftragskennzeichen 1 im intern erzeugten BIB
14	DML-Auftragskennzeichen 2 im intern erzeugten BIB
15	dynamisch vergebene UDS/SQL-interne Nummer der im DML angesprochenen Satzart bei BIBs alter Art
16	dynamisch vergebene UDS/SQL-interne Nummer von im DML genutztem Set bzw. Realm bei BIBs neuer Art
17-19	Status-Code des abgewickelten DML
20	Kennzeichen, ob der in Byte 17-19 ausgewiesene Status-Code mit dem in der BIB an den Anwender übergebenen übereinstimmt (’O’) oder nicht (’B’)
21	Kennzeichen, ob ein KDBS-Auftrag vorliegt
22	dynamisch vergebenes UDS/SQL-internes Datenbankkennzeichen
23-28	Subschema-Name

Byte 1-6: U02 CD

Byte	Bedeutung
9-12	dynamisch vergebene UDS/SQL-interne Transaktions-ID
13	DML-Auftragskennzeichen 1 im intern erzeugten BIB
14	DML-Auftragskennzeichen 2 im intern erzeugten BIB
15-16	dynamisch vergebene UDS/SQL-interne Nummer der im DML angesprochenen Satzart bei BIBs neuer Art
17-19	Status-Code des abgewickelten DML
20	Kennzeichen, ob der in Byte 17-19 ausgewiesene Status-Code mit dem in der BIB an den Anwender übergebenen übereinstimmt (’O’) oder nicht (’B’)
21	Kennzeichen, ob ein KDBS-Auftrag vorliegt
22	dynamisch vergebenes UDS/SQL-internes Datenbankkennzeichen
23-28	Subschema-Name

Byte 1-6: U01 CN

Byte	Bedeutung
13-20	Name der UDS/SQL-Konfiguration
21-24	Returncode des ENAMP-SVC zum Anschluss an den CUP

Byte 1-6: U01 DC

Byte	Bedeutung
13-20	Name der UDS/SQL-Konfiguration

Byte 1-6: U01 FN

Byte	Bedeutung
9-12	dynamisch vergebene UDS/SQL-interne Transaktions-ID

Byte 1-6: U01 PA

Byte	Bedeutung
9-32	24 Byte der von openUTM übergebenen Startparameter

Byte 1-6: U01 PB

Byte	Bedeutung
9-32	keine Zusatzinformation

Byte 1-6: U01 RB

Byte	Bedeutung
9-12	dynamisch vergebene UDS/SQL-interne Transaktions-ID
19-20	Anzahl der offenen und wiederherzustellenden Verarbeitungsketten

Byte 1-6: U03 RB

Byte	Bedeutung
9-12	dynamisch vergebene UDS/SQL-interne Transaktions_ID
13-15	UDS/SQL-interne Zustandsanzeigen der Session, des letzten Auftrags in der Transaktion und der wiederherzustellenden Verarbeitungskette
17-18	UDS/SQL-interne Nummer der wiederherzustellenden Verarbeitungskette
19-20	Anzahl der offenen Verarbeitungsketten
21-24	Lage der wiederherzustellenden BIB im Communication-Pool
27-32	Subschemaname

Byte 1-6: U01 SB

Byte	Bedeutung
9-12	dynamisch vergebene UDS/SQL-interne Transaktions-ID
13-16	Anzahl der offenen Verarbeitungsketten und damit Anzahl der zu sichernden BIBs

Byte 1-6: U01 SQ

Byte	Bedeutung
9-12	dynamisch vergebene UDS/SQL-interne Transaktions-ID
13-16	dynamisch vergebene UDS/SQL-interne ID des SQL-Vorgangs
17-20	SQL-Returncode des SQL-Auftrags
21-24	SQL-Returncode der 2. Schicht des SQL-Auftrags (operation)
25	interner SQL-Auftragscode
27-28	interner SQL-Fehlercode des Connections
29-30	interner Modul-Code des Connections bei Fehlern
31-32	modulinterne Fehlernummer

Byte 1-6: U01 ST

Byte	Bedeutung
13-16	UDS/SQL-interne RLOG-ID bzgl. der die Statusanfrage erfolgt
17-20	Session Section Nr bzgl. der die Statusanfrage erfolgt
21-22	modulinterne Fehlercodes

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)
Aufbauen und Umstrukturieren
Benutzerhandbuch

UDS/SQL (BS2000)
Datenbankbetrieb
Benutzerhandbuch

UDS/SQL (BS2000)
Entwerfen und Definieren
Benutzerhandbuch

UDS/SQL (BS2000)
Meldungen
Benutzerhandbuch

UDS/SQL (BS2000)
Sichern, Informieren und Reorganisieren
Benutzerhandbuch

UDS/SQL (BS2000)
Taschenbuch

UDS (BS2000)
Dialogsystem IQS
Benutzerhandbuch

UDS-KDBS (BS2000)
Kompatible Datenbankschnittstelle
Benutzerhandbuch

SQL für UDS/SQL
Sprachbeschreibung

BS2000 OSD/BC
Kommandos
Benutzerhandbuch

BS2000 OSD/BC
Einführung in die Systembetreuung
Benutzerhandbuch

BS2000 OSD/BC
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

BS2000 OSD/BC
Einführung in das DVS
Benutzerhandbuch

SDF (BS2000)
Dialogschnittstelle SDF
Benutzerhandbuch

SORT (BS2000)
Benutzerhandbuch

SPACEOPT (BS2000)
Optimierung und Reorganisation von Platten
Benutzerhandbuch

LMS (BS2000)
SDF-Format
Benutzerhandbuch

DSSM/SSCM
Verwaltung von Subsystemen in BS2000
Benutzerhandbuch

ARCHIVE (BS2000)
Benutzerhandbuch

DRV (BS2000)
Dual Recording by Volume
Benutzerhandbuch

HSMS / HSMS-SV (BS2000)
Hierarchisches Speicher Management System
Band 1: Funktionen, Verwaltung und Installation
Benutzerhandbuch

SECOS (BS2000)
Security Control System
Benutzerhandbuch

openNet Server (BS2000)
BCAM
Referenzhandbuch

DCAM (BS2000)
Programmschnittstellen
Beschreibung

DCAM (BS2000)
Makroaufrufe
Benutzerhandbuch

OMNIS/OMNIS-MENU (BS2000)
Funktionen und Kommandos
Benutzerhandbuch

OMNIS/OMNIS-MENU (BS2000)
Administration und Programmierung
Benutzerhandbuch

openUTM
Konzepte und Funktionen
Benutzerhandbuch

openUTM
Anwendungen programmieren mit KDCS für COBOL, C und C++
Benutzerhandbuch

openUTM
Anwendungen generieren
Benutzerhandbuch

openUTM
Anwendungen administrieren
Benutzerhandbuch

openUTM
Einsatz von openUTM-Anwendungen unter BS2000
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose (BS2000)
Benutzerhandbuch

COBOL2000 (BS2000)
COBOL-Compiler
Sprachbeschreibung

COBOL2000 (BS2000)
COBOL-Compiler
Benutzerhandbuch

COBOL85 (BS2000)
COBOL-Compiler
Beschreibung

COBOL85 (BS2000)
COBOL-Compiler
Benutzerhandbuch

CRTE (BS2000)
Common Runtime Environment
Benutzerhandbuch

DRIVE/WINDOWS (BS2000)
Programmiersystem
Benutzerhandbuch

DRIVE/WINDOWS (BS2000)
Programmiersprache
Sprachbeschreibung

DRIVE/WINDOWS (BS2000)
Lexikon der DRIVE-Anweisungen
Referenzhandbuch

DRIVE/WINDOWS (BS2000/SINIX)
Lexikon der DRIVE-SQL-Anweisungen für UDS
Referenzhandbuch

DAB (BS2000)
Disk Access Buffer
Benutzerhandbuch

XHCS (BS2000)
8-bit-Code- und Unicode-Unterstützung im BS2000
Benutzerhandbuch

Unicode im BS2000
Übersichtshandbuch

BS2000 OSD/BC
Softbooks Deutsch
CD-ROM

openSM2 (BS2000)
Software Monitor
Benutzerhandbuch

SNMP Management (BS2000)
Benutzerhandbuch

Stichwörter

E