

# ESQL-COBOL V3.0A

ESQL-COBOL für SESAM/SQL-Server

## **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.

Für Ihre Kommentare stehen Ihnen Fax-Formulare auf den letzten Seiten dieses Handbuchs zur Verfügung.

Dort finden Sie auch die Adressen der zuständigen Redaktion.

## **Zertifizierte Dokumentation nach DIN EN ISO 9001:2000**

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 9001:2000 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH  
[www.cognitas.de](http://www.cognitas.de)

## **Copyright und Handelsmarken**

Copyright © Fujitsu Siemens Computers GmbH 2006.

Alle Rechte vorbehalten.

Liefermöglichkeiten und technische Änderungen vorbehalten.

Alle verwendeten Hard- und Softwarenamen sind Handelsnamen und/oder Warenzeichen der jeweiligen Hersteller.

---

# Inhalt

<b>1</b>	<b>Einleitung</b> . . . . .	<b>7</b>
<b>1.1</b>	<b>Kurzbeschreibung des Produkts</b> . . . . .	<b>8</b>
<b>1.2</b>	<b>Zielsetzung und Zielgruppen des Handbuchs</b> . . . . .	<b>8</b>
<b>1.3</b>	<b>Konzept des Handbuchs</b> . . . . .	<b>9</b>
<b>1.4</b>	<b>Readme-Datei</b> . . . . .	<b>9</b>
<b>1.5</b>	<b>Änderungen gegenüber Vorgängerversionen</b> . . . . .	<b>10</b>
<b>1.6</b>	<b>Verwendete Darstellungsmittel</b> . . . . .	<b>11</b>
1.6.1	Darstellungsmittel für Anweisungen . . . . .	11
1.6.2	SDF-Syntaxdarstellung . . . . .	12
<b>2</b>	<b>SQL in COBOL-Programme einbetten</b> . . . . .	<b>27</b>
<b>2.1</b>	<b>ESQL-COBOL-Anwendung erstellen</b> . . . . .	<b>28</b>
<b>2.2</b>	<b>Bestandteile eines ESQL-COBOL-Programms</b> . . . . .	<b>29</b>
<b>2.3</b>	<b>Benutzervariablen</b> . . . . .	<b>30</b>
2.3.1	Benutzervariablen definieren . . . . .	30
2.3.2	Benutzervariablen in SQL-Anweisungen angeben . . . . .	37
2.3.2.1	Namen untergeordneter Datenfelder qualifizieren . . . . .	37
2.3.2.2	Vektoren ansprechen . . . . .	38
2.3.3	Indikatorvariablen . . . . .	40
2.3.3.1	Indikatorvariable definieren . . . . .	40
2.3.3.2	Indikatorvariable in einer SQL-Anweisung angeben . . . . .	41
2.3.3.3	Übertragung von Werten überprüfen . . . . .	42
2.3.3.4	NULL-Wert übertragen . . . . .	42
<b>2.4</b>	<b>Zuordnung von SQL- und COBOL-Datentypen</b> . . . . .	<b>43</b>
2.4.1	Zeichenkette fester Länge . . . . .	44
2.4.2	Zeichenkette variabler Länge . . . . .	45
2.4.3	Nationale Zeichenkette fester Länge . . . . .	47

2.4.4	Nationale Zeichenkette variabler Länge . . . . .	48
2.4.5	Kleine Ganzzahl . . . . .	50
2.4.6	Ganzzahl . . . . .	52
2.4.7	Festpunktzahl (gepackt) . . . . .	54
2.4.8	Festpunktzahl (ungepackt) . . . . .	55
2.4.9	Gleitpunktzahl mit einfacher Genauigkeit . . . . .	57
2.4.10	Gleitpunktzahl mit doppelter Genauigkeit . . . . .	58
2.4.11	Gleitpunktzahl . . . . .	58
2.4.12	Datum . . . . .	59
2.4.13	Uhrzeit . . . . .	61
2.4.14	Zeitstempel . . . . .	63
2.4.15	Vektoren . . . . .	65
<b>2.5</b>	<b>SQL-Anweisungen in einem ESQL-COBOL-Programm . . . . .</b>	<b>66</b>
<b>2.6</b>	<b>SQL-Kommentare in einem ESQL-COBOL-Programm . . . . .</b>	<b>68</b>
<b>2.7</b>	<b>Kommunikationsbereich . . . . .</b>	<b>69</b>
2.7.1	Aufbau des Kommunikationsbereichs . . . . .	69
2.7.2	Kommunikationsbereich bereitstellen . . . . .	72
2.7.3	Fehlerbehandlung und Erfolgskontrolle . . . . .	73
2.7.3.1	Programmablauf mit COBOL-Anweisungen steuern . . . . .	74
2.7.3.2	Programmablauf mit der SQL-Anweisung WHENEVER steuern . . . . .	75
<b>2.8</b>	<b>INCLUDE-Elemente . . . . .</b>	<b>77</b>
<b>3</b>	<b>ESQL-COBOL-Programm vorübersetzen . . . . .</b>	<b>79</b>
<b>3.1</b>	<b>ESQL-Precompiler aufrufen und steuern . . . . .</b>	<b>80</b>
3.1.1	Benötigte Bibliotheken und Dateien zuweisen . . . . .	80
3.1.2	Vorübersetzung mit Datenbankkontakt . . . . .	81
3.1.3	ESQL-Precompiler starten . . . . .	81
<b>3.2</b>	<b>ESQL-Precompiler-Optionen . . . . .</b>	<b>84</b>
3.2.1	Überblick über die ESQL-Precompiler-Optionen . . . . .	84
3.2.1.1	Eingabequellen festlegen . . . . .	84
3.2.1.2	Eigenschaften des ESQL-COBOL-Programms festlegen . . . . .	85
3.2.1.3	Vorübersetzung steuern . . . . .	86
3.2.2	Ausgabeziel für das erzeugte COBOL-Programm festlegen . . . . .	87
3.2.3	INCLUDE-Bibliotheken angeben . . . . .	89
3.2.4	Ausgabeziel für SQL-Bindelademodul festlegen . . . . .	90
3.2.5	Jobvariable angeben . . . . .	92
3.2.6	Vorübersetzung steuern . . . . .	93
3.2.7	Eingabequelle für den ESQL-Precompiler angeben . . . . .	96
3.2.8	Eigenschaften des ESQL-COBOL-Programms festlegen . . . . .	98

---

<b>3.3</b>	<b>Beendungsverhalten des ESQL-Precompiler</b> . . . . .	<b>105</b>
3.3.1	Überwachen des Beendungsverhaltens mit Jobvariablen . . . . .	105
3.3.2	Meldungen des ESQL-Precompiler . . . . .	106
3.3.3	Diagnoseunterlagen erstellen . . . . .	108
<b>4</b>	<b>COBOL-Programm übersetzen</b> . . . . .	<b>109</b>
<hr/>		
<b>5</b>	<b>ESQL-COBOL-Anwendung binden</b> . . . . .	<b>111</b>
<hr/>		
<b>6</b>	<b>ESQL-COBOL-Anwendung starten</b> . . . . .	<b>113</b>
<hr/>		
<b>7</b>	<b>ESQL-COBOL-Anwendung unter openUTM</b> . . . . .	<b>115</b>
<hr/>		
7.1	Sprachumfang unter openUTM . . . . .	115
7.2	UTM-Anwendung generieren . . . . .	116
7.3	UTM-Anwendung starten . . . . .	118
<b>8</b>	<b>Beispielprogramme</b> . . . . .	<b>119</b>
<hr/>		
8.1	Programm ABFRAGE . . . . .	120
8.2	Programm AENDERN . . . . .	125
8.3	Programm EINFUEGEN . . . . .	128
8.4	Programm LOESCHEN . . . . .	131
8.5	Programm DYNAMISCH . . . . .	135
<b>9</b>	<b>Meldungen des ESQL-COBOL-Systems</b> . . . . .	<b>141</b>
<hr/>		

<b>10</b>	<b>Anhang</b> . . . . .	<b>159</b>
<b>10.1</b>	<b>Mischbetrieb von SQL- und CALL-DML-Schnittstelle</b> . . . . .	<b>160</b>
<b>10.2</b>	<b>Beispieldatenbank</b> . . . . .	<b>160</b>
10.2.1	Schema AUFTRAGSVER . . . . .	160
10.2.1.1	Tabelle KUNDE . . . . .	161
10.2.1.2	Tabelle KONTAKT . . . . .	162
10.2.1.3	Tabelle AUFTRAG . . . . .	163
10.2.1.4	Tabelle LEISTUNG . . . . .	164
10.2.1.5	Tabelle AUFSTAT . . . . .	165
10.2.2	Schema TEILE . . . . .	166
10.2.2.1	Tabelle ARTIKEL . . . . .	166
10.2.2.2	Tabelle VERWENDUNG . . . . .	167
10.2.2.3	Tabelle LAGER . . . . .	168
10.2.2.4	Tabelle FARBTAB . . . . .	169
10.2.2.5	Tabelle TABTAB . . . . .	170
	<b>Fachwörter</b> . . . . .	<b>171</b>
	<b>Literatur</b> . . . . .	<b>177</b>
	Sonstige Literatur . . . . .	183
	<b>Stichwörter</b> . . . . .	<b>185</b>

---

# 1 Einleitung

ESQL-COBOL ist ein wichtiges Zusatzprodukt zum Datenbanksystem SESAM<sup>®</sup>/SQL-Server. Das Datenbanksystem SESAM/SQL-Server erfüllt durch seine Funktionen und seine Architekturmerkmale alle Anforderungen, die heute an einen leistungsfähigen Datenbankserver gestellt werden. Diese Eigenschaft drückt sich auch im Produktnamen SESAM/SQL-Server aus.

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

Im folgenden Kapitel finden Sie unter anderem:

- Kurzbeschreibung des Produkts ESQL-COBOL
- Zielsetzung und Zielgruppen des Handbuchs
- Konzept des Handbuchs
- Änderungen gegenüber der Vorgängerversion
- Verwendete Darstellungsmittel

## 1.1 Kurzbeschreibung des Produkts

ESQL-COBOL ermöglicht die Einbettung von SQL in die Wirtssprache COBOL.

Mit den in COBOL eingebetteten SQL-Anweisungen können Sie:

- SESAM/SQL-Datenbanken definieren, abfragen und ändern
- Datenbankverwaltungsfunktionen für SESAM/SQL-Datenbanken ausführen

### SQL

SQL (Structured Query Language) ist eine relationale Datenbanksprache, die von der ISO (International Organization for Standardization) mit der SQL-Norm ISO/IEC 9075:2003 standardisiert worden ist. Die Standardisierung von SQL ermöglicht die Erstellung von portablen SQL-Anwendungen.

ESQL-COBOL unterstützt die für den SQL-Standard verbindlichen Funktionen („mandatory features“) und zusätzlich einige optionale Funktionen vom SQL-Standard. Darüberhinaus bietet ESQL-COBOL leistungsfähige Erweiterungen zum SQL-Standard. Den unterstützten SQL-Sprachumfang beschreiben die SESAM/SQL-Handbücher „[SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen](#)“ [2] und „[SQL-Sprachbeschreibung Teil 2: Utilities](#)“ [3].

## 1.2 Zielsetzung und Zielgruppen des Handbuchs

Dieses Handbuch wendet sich an COBOL-Programmierer, die mit SQL-Anweisungen SESAM/SQL-Datenbanken definieren oder auf SESAM/SQL-Datenbanken zugreifen möchten.

Vorausgesetzt werden Grundkenntnisse im BS2000, SQL-Kenntnisse und Erfahrungen in der COBOL-Programmierung. Zur Generierung von Anwendungen unter openUTM sind UTM-Kenntnisse notwendig.

## 1.3 Konzept des Handbuchs

Dieses Handbuch beschreibt folgende Themen:

- Einbettung von SQL in ein COBOL-Programm
- Vorübersetzung des ESQL-COBOL-Programms mit dem ESQL-Precompiler
- Übersetzen eines COBOL-Programms
- Binden einer ESQL-COBOL-Anwendung
- Starten einer ESQL-COBOL-Anwendung
- ESQL-COBOL-Anwendungen unter UTM
- Beispielprogramme und Meldungen des ESQL-COBOL-Systems

Die Beispiele beziehen sich vorwiegend auf die im Anhang dargestellte Beispieldatenbank.

Das Handbuch enthält ein Fachwortverzeichnis mit den Definitionen der verwendeten Fachbegriffe.

Die gesuchten Informationen können Sie, außer über das Inhaltsverzeichnis, auch gezielt über das Stichwortverzeichnis und über Kolumnentitel nachschlagen. Literaturverweise finden Sie in Kurzform im Text. Der vollständige Titel der jeweiligen Druckschrift ist im Literaturverzeichnis aufgeführt.

## 1.4 Readme-Datei

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. der produktspezifischen Readme-Datei. Sie finden die Readme-Datei auf Ihrem BS2000-Rechner unter dem Dateinamen `SYSRME.produkt.version.sprache`. Die Benutzerkennung, unter der sich die Readme-Datei befindet, erfragen Sie bitte bei Ihrer zuständigen Systembetreuung. Die Readme-Datei können Sie mit dem Kommando `/SHOW-FILE` oder mit einem Editor ansehen oder auf einem Standarddrucker mit folgendem Kommando ausdrucken:

```
/PRINT-DOCUMENT dateiname, LINE-SPACING = *BY-EBCDIC-CONTROL
```

bei SPOOL -Versionen kleiner 3.0A:

```
/PRINT-FILE FILE-NAME = dateiname, LAYOUT-CONTROL =  
PARAMETERS(CONTROL-CHARACTERS = EBCDIC)
```

## 1.5 Änderungen gegenüber Vorgängerversionen

Dieser Abschnitt beschreibt die Änderungen des Handbuchs gegenüber der Vorgängerversion.

- Unterstützung der Zeichendarstellung mit UTF-16:
  - Datentyp NCHAR
  - Datentyp NVARCHAR

## 1.6 Verwendete Darstellungsmittel

In diesem Handbuch werden unterschiedliche Darstellungsmittel verwendet:

- Darstellungsmittel für Anweisungen, die nicht im SDF-Format vorliegen
- SDF-Syntaxdarstellung für Kommandos im SDF-Format

### 1.6.1 Darstellungsmittel für Anweisungen

Zur Darstellung der Anweisungen wird eine eigene Metasyntax verwendet. Die Bedeutung der einzelnen Sprachelemente ist nachfolgend in einer Tabelle zusammengefasst:

Sprachelement	Bedeutung	Beispiel
<i>kursiv</i>	Kursive Schrift kennzeichnet eine Variable, für die ein aktueller Wert einzusetzen ist.	<i>stufen_nr datenname</i>  Eingabebeispiel: 01 FIRMA
Schreibmaschinenschrift	Schreibmaschinenschrift wird in Beispielen verwendet.	EXEC SQL  Eingabebeispiel: EXEC SQL
{ }	Geschweifte Klammern schließen Alternativen ein. Alternative Ausdrücke sind in Klammern senkrecht untereinander aufgeführt. Genau ein Ausdruck ist auszuwählen. Die Klammer darf nicht angegeben werden.	$\left\{ \begin{matrix} X \\ A \\ 9 \end{matrix} \right\}$  Eingabebeispiel: 9
[ ]	In eckige Klammern eingeschlossene Ausdrücke dürfen weggelassen werden. Die Klammer darf nicht angegeben werden.	[INDICATOR]
... ,...  { },...	Wiederholungszeichen: Der unmittelbar vorhergehende Ausdruck darf - durch Leerzeichen bzw. Komma getrennt - mehrmals wiederholt werden. Der mit geschweiften Klammern eingeschlossene Ausdruck darf - durch Komma getrennt - mehrmals wiederholt werden.	<i>ausdruck, ...</i>  Eingabebeispiel: WOHNORT, GEHALT

Tabelle 1: Darstellungsmittel für COBOL-Anweisungen

## 1.6.2 SDF-Syntaxdarstellung

Bild 1 zeigt ein Beispiel für die Syntaxdarstellung eines Kommandos in einem Handbuch. Das Kommandoformat besteht aus einem Feld mit dem Kommandonamen. Anschließend werden alle Operanden mit den zulässigen Operandenwerten aufgelistet. Struktureinleitende Operandenwerte und die von ihnen abhängigen Operanden werden zusätzlich aufgelistet.

HELP-SDF	Kurzname: <b>HP</b> SDF
<b>GUIDANCE-MODE = <u>*NO</u> / *YES</b> <b>,SDF-COMMANDS = <u>*NO</u> / *YES</b> <b>,ABBREVIATION-RULES = <u>*NO</u> / *YES</b> <b>,GUIDED-DIALOG = <u>*YES</u> (...)</b> <b><u>*YES</u>(...)</b>   <b>SCREEN-STEPS = <u>*NO</u> / *YES</b> <b>,SPECIAL-FUNCTIONS = <u>*NO</u> / *YES</b> <b>,FUNCTION-KEYS = <u>*NO</u> / *YES</b> <b>,NEXT-FIELD = <u>*NO</u> / *YES</b> <b>,UNGUIDED-DIALOG = <u>*YES</u> (... ) / *NO</b> <b><u>*YES</u>(...)</b>   <b>SPECIAL-FUNCTIONS = <u>*NO</u> / *YES</b> <b>,FUNCTION-KEYS = <u>*NO</u> / *YES</b>	

Bild 1: Syntaxdarstellung des Benutzer-Kommandos HELP-SDF

Diese Syntaxbeschreibung basiert auf der SDF-Version 4.0A. Die Syntax der SDF-Kommando-/Anweisungssprache wird im folgenden in drei Tabellen erklärt.

Zu [Tabelle 2](#): *Metasyntax*

In den Kommando-/Anweisungsformaten werden bestimmte Zeichen und Darstellungsformen verwendet, deren Bedeutung in [Tabelle 2](#) erläutert wird.

Zu [Tabelle 3](#): *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 3](#) beschriebenen Datentypen.

Die Beschreibung der Datentypen gilt für alle Kommandos und Anweisungen. Deshalb werden bei den entsprechenden Operandenbeschreibungen nur noch Abweichungen von [Tabelle 3](#) erläutert.

Zu [Tabelle 4](#): Zusätze zu Datentypen

Zusätze zu Datentypen kennzeichnen weitere Eingabevorschriften für Datentypen. Die Zusätze schränken den Wertevorrat ein oder erweitern ihn. Im Handbuch werden folgende Zusätze in gekürzter Form dargestellt:

cat-id	cat
completion	compl
construction	constr
correction-state	corr
generation	gen
lower-case	low
manual-release	man
odd-possible	odd
path-completion	path-compl
separators	sep
underscore	under
user-id	user
version	vers
wildcards	wild

Für den Datentyp `integer` enthält [Tabelle 4](#) 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 4](#) erläutert.

## Metasyntax

Kennzeichnung	Bedeutung	Beispiele
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Schlüsselwörter. Schlüsselwörter für konstante Operandenwerte beginnen mit *	<b>HELP-SDF</b>  <b>SCREEN-STEPS = *NO</b>
<b>GROSSBUCHSTABEN</b> in Halbfett	Großbuchstaben in Halbfett kennzeichnen garantierte bzw. vorgeschlagene Abkürzungen der Schlüsselwörter.	<b>GUIDANCE-MODE = *YES</b>
=	Das Gleichheitszeichen verbindet einen Operandennamen mit den dazugehörigen Operandenwerten.	<b>GUIDANCE-MODE = *NO</b>
< >	Spitze Klammern kennzeichnen Variablen, deren Wertevorrat durch Datentypen und ihre Zusätze beschrieben wird (siehe Tabellen 3 und 4).	<b>SYNTAX-FILE = &lt;full-filename 1..54&gt;</b>
<u>Unterstreich</u>	Der Unterstrich kennzeichnet den Default-Wert eines Operanden.	<b>GUIDANCE-MODE = *NO</b>
/	Der Schrägstrich trennt alternative Operandenwerte.	<b>NEXT-FIELD = *NO / *YES</b>
(...)	Runde Klammern kennzeichnen Operandenwerte, die eine Struktur einleiten.	<b>,UNGUIDED-DIALOG = *YES (...)/ *NO</b>
[ ]	Eckige Klammern kennzeichnen struktureinleitende Operandenwerte, deren Angabe optional ist. Die nachfolgende Struktur kann ohne den einleitenden Operandenwert angegeben werden.	<b>SELECT = [*BY-ATTRIBUTES](...)</b>
Einrückung	Die Einrückung kennzeichnet die Abhängigkeit zu dem jeweils übergeordneten Operanden.	<b>,GUIDED-DIALOG = *YES (...)</b> <b>*YES(...)</b>   <b>SCREEN-STEPS = *NO / *YES</b>

Tabelle 2: Metasyntax

(Teil 1 von 2)

Kennzeichnung	Bedeutung	Beispiele
<p> </p> <p>,</p> <p>list-poss(n):</p>	<p>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.</p> <p>Das Komma steht vor weiteren Operanden der gleichen Strukturstufe.</p> <p>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.</p>	<pre>SUPPORT = *TAPE(...)       *TAPE(...)                 VOLUME = *<u>ANY</u>(...)           *<u>ANY</u>(...)             ...</pre> <p><b>GUIDANCE-MODE = *NO / *YES</b></p> <p><b>,SDF-COMMANDS = *NO / *YES</b></p> <p>list-poss: <b>*SAM / *ISAM</b></p> <p>list-poss(40): &lt;structured-name 1..30&gt;</p> <p>list-poss(256): <b>*OMF / *SYSLST(...)</b> /          &lt;full-filename 1..54&gt;</p>
<p>Kurzname:</p>	<p>Der darauf folgende Name ist ein garantierter Aliasname des Kommando- bzw. Anweisungsnamen.</p>	<p><b>HELP-SDF</b>      Kurzname: <b>HPSDF</b></p>

Tabelle 2: Metasyntax

(Teil 2 von 2)

## Datentypen

Datentyp	Zeichenvorrat	Besonderheiten
alphanum-name	A...Z 0...9 \$, #, @	
cat-id	A...Z 0...9	maximal 4 Zeichen; darf nicht mit der Zeichenfolge PUB beginnen
command-rest	beliebig	
composed-name	A...Z 0...9 \$, #, @ Bindestrich Punkt Katalogkennung	alphanumerische Zeichenfolge, die in mehrere durch Punkt oder Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann. Ist auch die Angabe eines Dateinamens möglich, so kann die Zeichenfolge mit einer Katalogkennung im Format :cat: beginnen (siehe Datentyp full-filename).
c-string	EBCDIC-Zeichen	ist in Hochkommata einzuschließen; der Buchstabe C kann vorangestellt werden; Hochkommata innerhalb des c-string müssen verdoppelt werden
date	0...9 Strukturkennzeichen: Bindestrich	Eingabeformat: jjjj-mm-tt  jjjj: Jahr; wahlweise 2- oder 4stellig mm: Monat tt: Tag
device	A...Z 0...9 Bindestrich	Zeichenfolge, die maximal 8 Zeichen lang ist und 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.
fixed	+, - 0...9 Punkt	Eingabeformat: [zeichen][ziffern].[ziffern]  [zeichen]: + oder - [ziffern]: 0...9  muss mindestens eine Ziffer, darf aber außer dem Vorzeichen maximal 10 Zeichen (0...9, Punkt) enthalten

Tabelle 3: Datentypen

(Teil 1 von 6)

Datentyp	Zeichenvorrat	Besonderheiten
full-filename	A...Z 0...9 \$, #, @ Bindestrich Punkt	Eingabeformat: $[:cat:][\$user.] \left\{ \begin{array}{l} \text{datei} \\ \text{datei(nr)} \\ \text{gruppe} \end{array} \right\}$ $\left. \begin{array}{l} \text{gruppe} \left\{ \begin{array}{l} (*abs) \\ (+rel) \\ (-rel) \end{array} \right\} \end{array} \right\}$ :cat: wahlfreie Angabe der Katalogkennung; Zeichenvorrat auf A...Z und 0...9 eingeschränkt; max. 4 Zeichen; ist in Doppelpunkte einzuschließen; voreingestellt ist die Katalogkennung, die der Benutzerkennung laut Eintrag im Benutzerkatalog zugeordnet ist. \$user. wahlfreie Angabe der Benutzerkennung; Zeichenvorrat ist A...Z, 0...9, \$, #, @; max. 8 Zeichen; darf nicht mit einer Ziffer beginnen; \$ und Punkt müssen angegeben werden; voreingestellt ist die eigene Benutzerkennung. \$. (Sonderfall) System-Standardkennung datei Datei- oder Jobvariablenname; kann durch Punkt in mehrere Teilnamen gegliedert sein: name <sub>1</sub> [.name <sub>2</sub> [...]] name <sub>i</sub> enthält keinen Punkt und darf nicht mit Bindestrich beginnen oder enden; datei ist max. 41 Zeichen lang, darf nicht mit \$ beginnen und muss mindestens ein Zeichen aus A...Z enthalten.

Tabelle 3: Datentypen

Datentyp	Zeichenvorrat	Besonderheiten
full-filename (Forts.)		<p>#datei (Sonderfall) @datei (Sonderfall) # oder @ als erstes Zeichen kennzeichnet je nach Systemgenerierung temporäre Dateien und Jobvariablen.</p> <p>datei(nr) Banddateiname nr: Versionsnummer; Zeichenvorrat ist A...Z, 0...9, \$, #, @. Klammern müssen angegeben werden.</p> <p>gruppe Name einer Dateigenerationsgruppe (Zeichenvorrat siehe unter "datei")</p> <p>gruppe <math>\left\{ \begin{array}{l} (*abs) \\ (+rel) \\ (-rel) \end{array} \right\}</math></p> <p>(*abs) absolute Generationsnummer (1..9999); * und Klammern müssen angegeben werden.</p> <p>(+rel) (-rel) relative Generationsnummer (0..99); Vorzeichen und Klammern müssen angegeben werden.</p>
integer	0...9, +, -	+ bzw. - kann nur erstes Zeichen sein (Vorzeichen).
name	A...Z 0...9 \$, #, @	darf nicht mit einer Ziffer beginnen.

Tabelle 3: Datentypen

(Teil 3 von 6)

Datentyp	Zeichenvorrat	Besonderheiten
partial-filename	A...Z 0...9 \$, #, @ Bindestrich Punkt	Eingabeformat: [:cat:][\${user.}][teilname.]  :cat: siehe full-filename \$user. siehe full-filename  teilname wahlfreie Angabe des gemeinsamen ersten Namensteils von Dateien und Dateigenerationsgruppen in der Form: name <sub>1</sub> . [name <sub>2</sub> . [...]] name <sub>i</sub> ; siehe full-filename. Das letzte Zeichen von teilname muss ein Punkt sein. Es muss mindestens einer der Teile :cat:, \$user. oder teilname angegeben werden.
posix-filename	A...Z 0...9 Sonderzeichen	Zeichenfolge, die maximal 255 Zeichen lang ist. Besteht entweder aus einem oder zwei Punkten, oder aus alphanumerischen Zeichen und Sonderzeichen; Sonderzeichen sind mit dem Zeichen \ zu entwerten. Nicht erlaubt ist das Zeichen /. Muss in Hochkommata eingeschlossen werden, wenn alternative Datentypen zulässig sind, Separatoren verwendet werden oder das erste Zeichen ? bzw. ! ist. Zwischen Groß- und Kleinschreibung wird unterschieden.
posix-pathname	A...Z 0...9 Sonderzeichen Strukturkennzeichen: Schrägstrich	Eingabeformat: [/]part <sub>1</sub> /.../part <sub>n</sub> wobei part <sub>i</sub> ein posix-filename ist; maximal 1024 Zeichen; muss in Hochkommata eingeschlossen werden, wenn alternative Datentypen zulässig sind, Separatoren verwendet werden oder das erste Zeichen ? bzw. ! ist.

Tabelle 3: Datentypen

(Teil 4 von 6)

Datentyp	Zeichenvorrat	Besonderheiten
product-version	A...Z 0...9 Punkt Hochkomma	<p>Eingabeformat: <math>[[C]' ][V][n]n.nann[' ]</math></p> <div style="margin-left: 150px;"> <math>\begin{array}{c}   \\   \\ \text{Korrekturstand} \\ \text{Freigabestand} \end{array}</math> </div> <p>wobei n eine Ziffer und a ein Buchstabe ist. Freigabe- und Korrekturstand müssen angegeben werden, wenn product-version keinen Zusatz enthält (siehe <a href="#">Tabelle 4</a>, Zusatz without-corr bzw. without-man). product-version kann in Hochkommata eingeschlossen werden, wobei der Buchstabe C vorangestellt werden kann. Die Versionsangabe kann mit dem Buchstaben V beginnen.</p>
structured-name	A...Z 0...9 \$, #, @ Bindestrich	<p>alphanumerische Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A...Z oder \$, #, @</p>
text	beliebig	Das Eingabeformat ist den jeweiligen Operandenbeschreibungen zu entnehmen.
time	0...9 Strukturkennzeichen: Doppelpunkt	<p>Angabe einer Tageszeit</p> <p>Eingabeformat: <math>\left. \begin{array}{l} hh:mm:ss \\ hh:mm \\ hh \end{array} \right\}</math></p> <p>hh:    Stunden mm:    Minuten ss:    Sekunden <math>\left. \right\}</math> führende Nullen können weggelassen werden</p>
vsn	<p>a) A...Z 0...9</p> <p>b) A...Z 0...9 \$, #, @</p>	<p>a) Eingabeformat: pvsid.folgenummer max. 6 Zeichen; pvsid:            2-4 Zeichen; Eingabe von PUB nicht erlaubt folgenummer:  1-3 Zeichen</p> <p>b) max. 6 Zeichen; PUB darf vorangestellt werden, dann dürfen jedoch nicht \$,#,@ folgen.</p>

Tabelle 3: Datentypen

(Teil 5 von 6)

Datentyp	Zeichenvorrat	Besonderheiten
x-string	Sedezimal: 00...FF	ist in Hochkommata einzuschließen; der Buchstabe X muss vorangestellt werden; die Anzahl der Zeichen darf ungerade sein.
x-text	Sedezimal: 00...FF	ist nicht in Hochkommata einzuschließen; der Buchstabe X darf nicht vorangestellt werden; die Anzahl der Zeichen darf ungerade sein.

Tabelle 3: Datentypen

(Teil 6 von 6)

### Zusätze zu Datentypen

Zusatz	Bedeutung
x..y <i>unit</i>	<p>a) beim Datentyp integer: Intervallangabe</p> <p>x     Mindestwert, der für integer erlaubt ist. x ist eine ganze Zahl, die mit einem Vorzeichen versehen werden darf.</p> <p>y     Maximalwert, der für integer erlaubt ist. y ist eine ganze Zahl, die mit einem Vorzeichen versehen werden darf.</p> <p><i>unit</i>     nur bei Datentyp integer: zusätzliche Einheiten. Folgende Angaben werden verwendet:</p> <p><i>days</i>     <i>byte</i> <i>hours</i>     <i>2Kbyte</i> <i>minutes</i>   <i>4Kbyte</i> <i>seconds</i>   <i>Mbyte</i></p> <p>b) bei den übrigen Datentypen: Längenangabe</p> <p>x     Mindestlänge für den Operandenwert; x ist eine ganze Zahl.</p> <p>y     Maximallänge für den Operandenwert; y ist eine ganze Zahl.</p> <p>x = y   Der Operandenwert muss genau die Länge x haben.</p>
with	Erweitert die Angabemöglichkeiten für einen Datentyp.
-compl	Bei Angaben zu dem Datentyp date ergänzt SDF zweistellige Jahresangaben der Form jj-mm-tt zu:
	<p>20jj-mm-tt     falls jj &lt; 60</p> <p>19jj-mm-tt     falls jj ≥ 60</p>
-low	Groß- und Kleinschreibung wird unterschieden.
-under	Erlaubt Unterstriche '_' beim Datentyp name.

Tabelle 4: Zusätze zu Datentypen

(Teil 1 von 6)

Zusatz	Bedeutung
with (Forts.)	
-wild(n)	Teile eines Namens dürfen durch die folgenden Platzhalter ersetzt werden. n bezeichnet die maximale Eingabelänge bei Verwendung von Platzhaltern. Mit Einführung der Datentypen posix-filename und posix-pathname werden neben den bisher im BS2000 üblichen Platzhaltern auch Platzhalter aus der UNIX-Welt (nachfolgend POSIX-Platzhalter genannt) akzeptiert. Innerhalb einer Musterzeichenfolge sollten entweder nur BS2000- oder nur POSIX-Platzhalter verwendet werden. Bei den Datentypen posix-filename und posix-pathname sind nur POSIX-Platzhalter erlaubt. Ist eine Musterzeichenfolge mehrdeutig auf einen String abbildbar, gilt der erste Treffer.
BS2000-Platzhalter	Bedeutung
*	Ersetzt eine beliebige, auch leere Zeichenfolge. Ein * an erster Stelle muss verdoppelt werden, sofern dem * weitere Zeichen folgen und die eingegebene Zeichenfolge nicht mindestens einen weiteren Platzhalter enthält.
Punkt am Ende	Teilqualifizierte Angabe eines Namens. Entspricht implizit der Zeichenfolge ".*", d.h. nach dem Punkt folgt mindestens ein beliebiges Zeichen.
/	Ersetzt genau ein beliebiges Zeichen.
<s <sub>x</sub> :s <sub>y</sub> >	Ersetzt eine Zeichenfolge, für die gilt: <ul style="list-style-type: none"> <li>– sie ist mindestens so lang wie die kürzeste Zeichenfolge (s<sub>x</sub> oder s<sub>y</sub>)</li> <li>– sie ist höchstens so lang wie die längste Zeichenfolge (s<sub>x</sub> oder s<sub>y</sub>)</li> <li>– sie liegt in der alphabetischen Sortierung zwischen s<sub>x</sub> und s<sub>y</sub>; Zahlen werden hinter Buchstaben sortiert (A...Z 0...9)</li> <li>– s<sub>x</sub> darf auch die leere Zeichenfolge sein, die in der alphabetischen Sortierung an erster Stelle steht</li> <li>– s<sub>y</sub> darf auch die leere Zeichenfolge sein, die an dieser Stelle für die Zeichenfolge mit der höchst möglichen Codierung steht (enthält nur die Zeichen 'FF')</li> </ul>
<s <sub>1</sub> ,...>	Ersetzt alle Zeichenfolgen, auf die eine der mit s angegebenen Zeichenkombinationen zutrifft. s kann auch die leere Zeichenfolge sein. Jede Zeichenfolge s kann auch eine Bereichsangabe "s <sub>x</sub> :s <sub>y</sub> " sein (siehe oben).

Tabelle 4: Zusätze zu Datentypen

(Teil 2 von 6)

Zusatz	Bedeutung														
with-wild(n) (Forts.)	<p>-s</p> <p>Ersetzt alle Zeichenfolgen, die der angegebenen Zeichenfolge s nicht entsprechen. Das Minuszeichen darf nur am Beginn der Zeichenfolge stehen.</p> <p>Innerhalb der Datentypen full-filename bzw. partial-filename kann die negierte Zeichenfolge -s genau einmal verwendet werden, d.h., -s kann einen der drei Namensteile cat, user oder datei ersetzen.</p> <p>Platzhalter sind in Generations- und Versionsangaben von Dateinamen nicht erlaubt. In Benutzerkennungen ist die Angabe von Platzhaltern der Systemverwaltung vorbehalten. Platzhalter können nicht die Begrenzer der Namensteile cat (Doppelpunkte) und user (\$ und Punkt) ersetzen.</p> <table border="1"> <thead> <tr> <th>POSIX-Platzhalter</th> <th>Bedeutung</th> </tr> </thead> <tbody> <tr> <td>*</td> <td>Ersetzt eine beliebige, auch leere Zeichenfolge. Ein * an erster Stelle muss verdoppelt werden, sofern dem * weitere Zeichen folgen und die eingegebene Zeichenfolge nicht mindestens einen weiteren Platzhalter enthält.</td> </tr> <tr> <td>?</td> <td>Ersetzt genau ein beliebiges Zeichen. Ist als erstes Zeichen außerhalb von Hochkommata nicht zulässig.</td> </tr> <tr> <td>[c<sub>x</sub>-c<sub>y</sub>]</td> <td>Ersetzt genau ein Zeichen aus dem Bereich c<sub>x</sub> und c<sub>y</sub> einschließlich der Bereichsgrenzen. c<sub>x</sub> und c<sub>y</sub> müssen einfache Zeichen sein.</td> </tr> <tr> <td>[s]</td> <td>Ersetzt genau ein Zeichen aus der Zeichenfolge s. Die Ausdrücke [c<sub>x</sub>-c<sub>y</sub>] und [s] können kombiniert werden zu [s<sub>1</sub>c<sub>1</sub>-c<sub>2</sub>s<sub>2</sub>]</td> </tr> <tr> <td>[!c<sub>x</sub>-c<sub>y</sub>]</td> <td>Ersetzt genau ein Zeichen, das nicht in dem Bereich c<sub>x</sub> und c<sub>y</sub> einschließlich der Bereichsgrenzen enthalten ist. c<sub>x</sub> und c<sub>y</sub> müssen einfache Zeichen sein. Die Ausdrücke [!c<sub>x</sub>-c<sub>y</sub>] und [!s] können kombiniert werden zu [!s<sub>1</sub>c<sub>1</sub>-c<sub>2</sub>s<sub>2</sub>]</td> </tr> <tr> <td>[!s]</td> <td>Ersetzt genau ein Zeichen, das nicht in der Zeichenfolge s enthalten ist. Die Ausdrücke [!s] und [!cx-cy] können kombiniert werden zu [!s<sub>1</sub>c<sub>1</sub>-c<sub>2</sub>s<sub>2</sub>]</td> </tr> </tbody> </table>	POSIX-Platzhalter	Bedeutung	*	Ersetzt eine beliebige, auch leere Zeichenfolge. Ein * an erster Stelle muss verdoppelt werden, sofern dem * weitere Zeichen folgen und die eingegebene Zeichenfolge nicht mindestens einen weiteren Platzhalter enthält.	?	Ersetzt genau ein beliebiges Zeichen. Ist als erstes Zeichen außerhalb von Hochkommata nicht zulässig.	[c <sub>x</sub> -c <sub>y</sub> ]	Ersetzt genau ein Zeichen aus dem Bereich c <sub>x</sub> und c <sub>y</sub> einschließlich der Bereichsgrenzen. c <sub>x</sub> und c <sub>y</sub> müssen einfache Zeichen sein.	[s]	Ersetzt genau ein Zeichen aus der Zeichenfolge s. Die Ausdrücke [c <sub>x</sub> -c <sub>y</sub> ] und [s] können kombiniert werden zu [s <sub>1</sub> c <sub>1</sub> -c <sub>2</sub> s <sub>2</sub> ]	[!c <sub>x</sub> -c <sub>y</sub> ]	Ersetzt genau ein Zeichen, das nicht in dem Bereich c <sub>x</sub> und c <sub>y</sub> einschließlich der Bereichsgrenzen enthalten ist. c <sub>x</sub> und c <sub>y</sub> müssen einfache Zeichen sein. Die Ausdrücke [!c <sub>x</sub> -c <sub>y</sub> ] und [!s] können kombiniert werden zu [!s <sub>1</sub> c <sub>1</sub> -c <sub>2</sub> s <sub>2</sub> ]	[!s]	Ersetzt genau ein Zeichen, das nicht in der Zeichenfolge s enthalten ist. Die Ausdrücke [!s] und [!cx-cy] können kombiniert werden zu [!s <sub>1</sub> c <sub>1</sub> -c <sub>2</sub> s <sub>2</sub> ]
POSIX-Platzhalter	Bedeutung														
*	Ersetzt eine beliebige, auch leere Zeichenfolge. Ein * an erster Stelle muss verdoppelt werden, sofern dem * weitere Zeichen folgen und die eingegebene Zeichenfolge nicht mindestens einen weiteren Platzhalter enthält.														
?	Ersetzt genau ein beliebiges Zeichen. Ist als erstes Zeichen außerhalb von Hochkommata nicht zulässig.														
[c <sub>x</sub> -c <sub>y</sub> ]	Ersetzt genau ein Zeichen aus dem Bereich c <sub>x</sub> und c <sub>y</sub> einschließlich der Bereichsgrenzen. c <sub>x</sub> und c <sub>y</sub> müssen einfache Zeichen sein.														
[s]	Ersetzt genau ein Zeichen aus der Zeichenfolge s. Die Ausdrücke [c <sub>x</sub> -c <sub>y</sub> ] und [s] können kombiniert werden zu [s <sub>1</sub> c <sub>1</sub> -c <sub>2</sub> s <sub>2</sub> ]														
[!c <sub>x</sub> -c <sub>y</sub> ]	Ersetzt genau ein Zeichen, das nicht in dem Bereich c <sub>x</sub> und c <sub>y</sub> einschließlich der Bereichsgrenzen enthalten ist. c <sub>x</sub> und c <sub>y</sub> müssen einfache Zeichen sein. Die Ausdrücke [!c <sub>x</sub> -c <sub>y</sub> ] und [!s] können kombiniert werden zu [!s <sub>1</sub> c <sub>1</sub> -c <sub>2</sub> s <sub>2</sub> ]														
[!s]	Ersetzt genau ein Zeichen, das nicht in der Zeichenfolge s enthalten ist. Die Ausdrücke [!s] und [!cx-cy] können kombiniert werden zu [!s <sub>1</sub> c <sub>1</sub> -c <sub>2</sub> s <sub>2</sub> ]														

Tabelle 4: Zusätze zu Datentypen

Zusatz	Bedeutung										
with (Forts.) -constr	<p>Angabe einer Konstruktionszeichenfolge, die angibt, wie aus einer zuvor angegebenen Auswahlzeichenfolge mit Musterzeichen (siehe with-wild) neue Namen zu bilden sind.</p> <p>Die Konstruktionszeichenfolge kann aus konstanten Zeichenfolgen und Musterzeichen bestehen. Ein Musterzeichen wird durch diejenige Zeichenfolge ersetzt, die durch das entsprechende Musterzeichen in der Auswahlzeichenfolge ausgewählt wird.</p> <p>Folgende Platzhalter können zur Konstruktionsangabe verwendet werden:</p> <table border="1"> <thead> <tr> <th>Platzhalter</th> <th>Bedeutung</th> </tr> </thead> <tbody> <tr> <td>*</td> <td>Entspricht der Zeichenfolge, die durch den Platzhalter * in der Auswahlzeichenfolge ausgewählt wird.</td> </tr> <tr> <td>Punkt am Ende</td> <td>Entspricht der teilqualifizierten Angabe eines Namens in der Auswahlzeichenfolge. Entspricht der Zeichenfolge, die durch den Punkt am Ende der Auswahlzeichenfolge ausgewählt wird.</td> </tr> <tr> <td>/ oder ?</td> <td>Entspricht dem Zeichen, das durch den Platzhalter / oder ? in der Auswahlzeichenfolge ausgewählt wird.</td> </tr> <tr> <td>&lt;n&gt;</td> <td>Entspricht der Zeichenfolge, die durch den n-ten Platzhalter in der Auswahlzeichenfolge ausgewählt wird; n = &lt;integer&gt;</td> </tr> </tbody> </table> <p>Zuordnung der Platzhalter zu entsprechenden Platzhaltern in der Auswahlzeichenfolge: In der Auswahlzeichenfolge werden alle Platzhalter von links nach rechts aufsteigend numeriert (globaler Index). Gleiche Platzhalter in der Auswahlzeichenfolge werden zusätzlich von links nach rechts aufsteigend numeriert (platzhalter-spezifischer Index). In der Konstruktionsangabe können Platzhalter auf zwei, sich gegenseitig ausschließende Arten angegeben werden:</p> <ol style="list-style-type: none"> <li>1. Platzhalter werden über den globalen Index angegeben: &lt;n&gt;</li> <li>2. Angabe desselben Platzhalters, wobei die Ersetzung gemäß dem platzhalter-spezifischen Index entsprechend erfolgt: z.B. der zweite "/" entspricht der Zeichenfolge, die durch den zweiten "/" in der Auswahlzeichenfolge ausgewählt wird.</li> </ol>	Platzhalter	Bedeutung	*	Entspricht der Zeichenfolge, die durch den Platzhalter * in der Auswahlzeichenfolge ausgewählt wird.	Punkt am Ende	Entspricht der teilqualifizierten Angabe eines Namens in der Auswahlzeichenfolge. Entspricht der Zeichenfolge, die durch den Punkt am Ende der Auswahlzeichenfolge ausgewählt wird.	/ oder ?	Entspricht dem Zeichen, das durch den Platzhalter / oder ? in der Auswahlzeichenfolge ausgewählt wird.	<n>	Entspricht der Zeichenfolge, die durch den n-ten Platzhalter in der Auswahlzeichenfolge ausgewählt wird; n = <integer>
Platzhalter	Bedeutung										
*	Entspricht der Zeichenfolge, die durch den Platzhalter * in der Auswahlzeichenfolge ausgewählt wird.										
Punkt am Ende	Entspricht der teilqualifizierten Angabe eines Namens in der Auswahlzeichenfolge. Entspricht der Zeichenfolge, die durch den Punkt am Ende der Auswahlzeichenfolge ausgewählt wird.										
/ oder ?	Entspricht dem Zeichen, das durch den Platzhalter / oder ? in der Auswahlzeichenfolge ausgewählt wird.										
<n>	Entspricht der Zeichenfolge, die durch den n-ten Platzhalter in der Auswahlzeichenfolge ausgewählt wird; n = <integer>										

Tabelle 4: Zusätze zu Datentypen

(Teil 4 von 6)

Zusatz	Bedeutung
with-constr (Forts.)	<p>Bei Konstruktionsangaben sind folgende Regeln zu beachten:</p> <ul style="list-style-type: none"> <li>– Die Konstruktionsangabe muss mindestens einen Platzhalter der Auswahlzeichenfolge enthalten.</li> <li>– Ist die Anzahl gleicher Platzhalter größer als in der Auswahlzeichenfolge, muss die Index-Schreibweise gewählt werden.</li> <li>– Soll die Zeichenkette, die der Platzhalter &lt;...&gt; bzw. [...] auswählt, in der Konstruktionsangabe verwendet werden, muss die Index-Schreibweise gewählt werden.</li> <li>– Die Index-Schreibweise muss gewählt werden, wenn die Zeichenkette, die der Platzhalter "*" bezeichnet, vervielfacht werden soll: Statt "***" muss z.B. "&lt;n&gt;&lt;n&gt;" angegeben werden.</li> <li>– Der Platzhalter * kann auch die leere Zeichenkette sein. Insbesondere ist zu beachten, dass bei mehreren Sternen in Folge (auch mit weiteren Platzhaltern) nur der letzte Stern eine nicht leere Zeichenfolge sein kann: z.B. bei "*****" oder "*//*".</li> <li>– Aus der Konstruktionsangabe sollten gültige Namen entstehen. Darauf ist sowohl bei der Auswahlangabe als auch bei der Konstruktionsangabe zu achten.</li> <li>– Abhängig von der Konstruktionsangabe können aus unterschiedlichen Namen, die in der Auswahlangabe ausgewählt werden, identische Namen gebildet werden:                      z.B. "A/*" wählt die Namen "A1" und "A2" aus; die Konstruktionsangabe "B*" erzeugt für beide Namen denselben neuen Namen "B".                      Um dies zu vermeiden, sollten in der Konstruktionsangabe alle Platzhalter der Auswahlangabe mindestens einmal verwendet werden.</li> <li>– Wird die Auswahlzeichenfolge mit einem Punkt abgeschlossen, so muss auch die Konstruktionsangabe mit einem Punkt enden (und umgekehrt).</li> </ul>

Tabelle 4: Zusätze zu Datentypen

(Teil 5 von 6)

Zusatz	Bedeutung																				
with-constr (Forts.)	Beispiele:																				
	<table border="1"> <thead> <tr> <th>Auswahlmuster</th> <th>Auswahl</th> <th>Konstruktionsmuster</th> <th>neuer Name</th> </tr> </thead> <tbody> <tr> <td>A//*</td> <td>AB1 AB2 A.B.C</td> <td>D&lt;3&gt;&lt;2&gt;</td> <td>D1 D2 D.CB</td> </tr> <tr> <td>C.&lt;A:C&gt;/&lt;D,F&gt;</td> <td>C.AAD C.ABD C.BAF C.BBF</td> <td>G.&lt;1&gt;.&lt;3&gt;.XY&lt;2&gt;</td> <td>G.A.D.XYA G.A.D.XYB G.B.F.XYA G.B.F.XYB</td> </tr> <tr> <td>C.&lt;A:C&gt;/&lt;D,F&gt;</td> <td>C.AAD C.ABD C.BAF C.BBF</td> <td>G.&lt;1&gt;.&lt;2&gt;.XY&lt;2&gt;</td> <td>G.A.A.XYA G.A.B.XYB G.B.A.XYA G.B.B.XYB</td> </tr> <tr> <td>A//B</td> <td>ACDB ACEB AC.B A.CB</td> <td>G/XY/</td> <td>GCXYD GCXYE GCXY. G.XYC</td> </tr> </tbody> </table>	Auswahlmuster	Auswahl	Konstruktionsmuster	neuer Name	A//*	AB1 AB2 A.B.C	D<3><2>	D1 D2 D.CB	C.<A:C>/<D,F>	C.AAD C.ABD C.BAF C.BBF	G.<1>.<3>.XY<2>	G.A.D.XYA G.A.D.XYB G.B.F.XYA G.B.F.XYB	C.<A:C>/<D,F>	C.AAD C.ABD C.BAF C.BBF	G.<1>.<2>.XY<2>	G.A.A.XYA G.A.B.XYB G.B.A.XYA G.B.B.XYB	A//B	ACDB ACEB AC.B A.CB	G/XY/	GCXYD GCXYE GCXY. G.XYC
	Auswahlmuster	Auswahl	Konstruktionsmuster	neuer Name																	
	A//*	AB1 AB2 A.B.C	D<3><2>	D1 D2 D.CB																	
	C.<A:C>/<D,F>	C.AAD C.ABD C.BAF C.BBF	G.<1>.<3>.XY<2>	G.A.D.XYA G.A.D.XYB G.B.F.XYA G.B.F.XYB																	
C.<A:C>/<D,F>	C.AAD C.ABD C.BAF C.BBF	G.<1>.<2>.XY<2>	G.A.A.XYA G.A.B.XYB G.B.A.XYA G.B.B.XYB																		
A//B	ACDB ACEB AC.B A.CB	G/XY/	GCXYD GCXYE GCXY. G.XYC																		
1) Punkt am Ende des Namens kann Namenskonvention widersprechen (z.B bei vollqualifizierten Dateinamen)																					
without	Schränkt die Angabemöglichkeiten für einen Datentyp ein.																				
-cat	Die Angabe einer Katalogkennung ist nicht erlaubt.																				
-corr	Eingabeformat: [[C]' ][V][n]n.na[ ] Angaben zum Datentyp product-version dürfen den Korrekturstand nicht enthalten.																				
-gen	Die Angabe einer Dateigeneration oder Dateigenerationsgruppe ist nicht erlaubt.																				
-man	Eingabeformat: [[C]' ][V][n]n.n[ ] Angaben zum Datentyp product-version dürfen weder Freigabe- noch Korrekturstand enthalten.																				
-odd	Der Datentyp x-text erlaubt nur eine gerade Anzahl von Zeichen.																				
-sep	Beim Datentyp text ist die Angabe der folgenden Trennzeichen nicht erlaubt: ; = ( ) < > ? (also Strichpunkt, Gleichheitszeichen, runde Klammer auf und zu, Größerzeichen, Kleinerzeichen und Leerzeichen)																				
-user	Die Angabe einer Benutzerkennung ist nicht erlaubt.																				
-vers	Die Angabe der Version (siehe "datei(nr)") ist bei Banddateien nicht erlaubt.																				

Tabelle 4: Zusätze zu Datentypen

(Teil 6 von 6)

---

## 2 SQL in COBOL-Programme einbetten

In diesem Kapitel erfahren Sie, wie Sie eine ESQL-COBOL-Anwendung erstellen. Das Kapitel enthält:

- ESQL-COBOL-Anwendung erstellen
- Bestandteile eines ESQL-COBOL-Programms
- Benutzervariablen
- Zuordnung von SQL- und COBOL-Datentypen
- SQL-Anweisungen in einem ESQL-COBOL-Programm
- SQL-Kommentare in einem ESQL-COBOL-Programm
- Kommunikationsbereich
- INCLUDE-Elemente

## 2.1 ESQL-COBOL-Anwendung erstellen

Die folgende Abbildung zeigt, wie Sie aus einem ESQL-COBOL-Programm eine ablauffähige ESQL-COBOL-Anwendung erstellen.

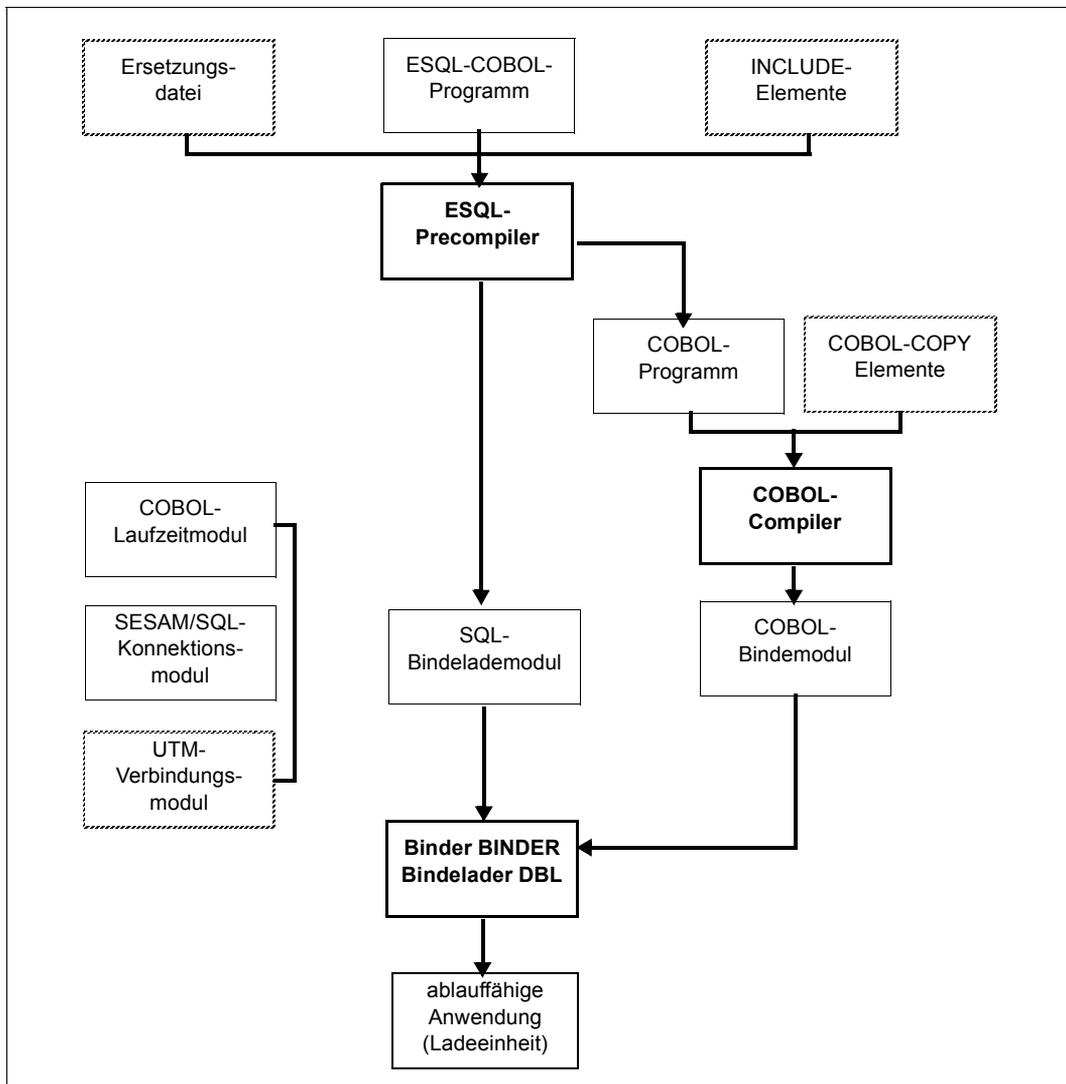


Bild 2: ESQL-COBOL-Anwendung erstellen

### **Vorübersetzen**

Das ESQL-COBOL-Programm wird mit dem ESQL-Precompiler vorübersetzt. Enthält das ESQL-COBOL-Programm die SQL-Anweisung INCLUDE, werden die entsprechenden INCLUDE-Elemente eingebunden. Zur Umstellung oder Portierung von ESQL-COBOL-Anwendungen können Ersetzungsdateien verwendet werden. Diese erleichtern die Ersetzung von Namen, Namen in Anführungszeichen und Schlüsselwörtern. Ist das ESQL-COBOL-Programm fehlerfrei, erzeugt der ESQL-Precompiler ein SQL-Bindelademodul (LLM) und ein COBOL-Programm, das keine SQL-Anweisungen mehr enthält.

### **Übersetzen**

Das erzeugte COBOL-Programm wird mit dem COBOL-Compiler COBOL2000 übersetzt. Ist das erzeugte COBOL-Programm fehlerfrei, erzeugt der COBOL-Compiler daraus ein COBOL-Bindemodul.

### **Binden**

Die erzeugten Module, das COBOL-Laufzeitmodul, das SESAM/SQL-Konnektionsmodul und ggf. UTM-Verbindungsmodule werden mit dem Binder BINDER und dem Bindelader DBL zu einer ablauffähigen ESQL-COBOL-Anwendung in Form einer Ladeinheit zusammengebunden.

## **2.2 Bestandteile eines ESQL-COBOL-Programms**

Ein ESQL-COBOL-Programm besteht zusätzlich zu COBOL-Sprachelementen aus folgenden Teilen:

- Definition von Benutzervariablen
- SQL-Anweisungen und -Kommentare
- Kommunikationsbereich

Die folgenden Abschnitte beschreiben, wie Sie die einzelnen Elemente in ein COBOL-Programm einbetten und welche Regeln Sie dabei beachten müssen.

## 2.3 Benutzervariablen

Eine Benutzervariable ist ein COBOL-Datenfeld, das Sie in einer eingebetteten SQL-Anweisung verwenden können. Mit einer Benutzervariablen können Sie Werte aus der Datenbank in das ESQL-COBOL-Programm übertragen und dort weiterverarbeiten. Umgekehrt können Sie mit einer Benutzervariablen auch Daten in die Datenbank übertragen und Werte bereitstellen, die in Berechnungen benötigt werden. Sie können eine Benutzervariable auch als Indikatorvariable verwenden. Eine Indikatorvariable ist eine spezielle Benutzervariable, die zur Überwachung der Datenübertragung und zur Übertragung des NULL-Werts dient. In welchen SQL-Anweisungen Benutzervariablen verwendet werden können oder müssen, beschreibt das Handbuch „[SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen](#)“ [2].

### 2.3.1 Benutzervariablen definieren

Eine Benutzervariable muss im Text des ESQL-COBOL-Programms vor ihrer Verwendung in einer SQL-Anweisung definiert sein. Bei geschachtelten Programmen muss die Definition der Benutzervariablen bei jeder Verwendung der Benutzervariablen gemäß COBOL-Regeln sichtbar sein (siehe auch [Abschnitt „Benutzervariablen definieren“ auf Seite 35](#)). Bei einer Benutzervariable, die in einer DECLARE CURSOR-Anweisung verwendet wird, muss die Definition bei jeder OPEN CURSOR-Anweisung für diesen Cursor sichtbar sein.

Sie definieren Benutzervariablen innerhalb einer DECLARE SECTION in der DATA DIVISION. Sie können beliebig viele DECLARE SECTIONs definieren. DECLARE SECTIONs dürfen jedoch nicht geschachtelt werden. Für die Definition von Benutzervariablen gelten die COBOL-Konventionen zur Definition von Datenfeldern sowie SESAM/SQL-spezifische Regeln. Zusätzlich werden Datentypen unterstützt, die den SQL-Datentypen VARCHAR, NVARCHAR, DATE, TIME(3) und TIMESTAMP(3) entsprechen. Die Beschreibung dieser Datentypen und der SESAM/SQL-spezifischen Regeln finden Sie im [Abschnitt „Zuordnung von SQL- und COBOL-Datentypen“ auf Seite 43](#). Eine ausführliche Beschreibung der Definition von Datenfeldern in COBOL finden Sie im COBOL2000 Handbuch „[Sprachbeschreibung](#)“ [16].

---

```
EXEC SQL BEGIN DECLARE SECTION END-EXEC
```

```
    benutzervariablen_def...
```

```
EXEC SQL END DECLARE SECTION END-EXEC
```

---

```

benutzervariablen_def : =
    stufennummer [ { datenname } ] [ {
        PICTURE-Klausel
        BASED-Klausel
        USAGE-Klausel
        GROUP-USAGE-Klausel
        SIGN-Klausel
        VALUE-Klausel
        SYNC-Klausel
        OCCURS-Klausel
        EXTERNAL-Klausel
        GLOBAL-Klausel
        REDEFINES-Klausel
        VARCHAR
        NVARCHAR
        DATE
        TIME(3)
        TIMESTAMP(3)
    } ... ]

```

Enthält die Spalte 7 einer Zeile innerhalb der DECLARE SECTION ein anderes Zeichen als ein Leerzeichen, interpretiert der ESQL-Precompiler die Zeile als Kommentarzeile. Innerhalb einer DECLARE SECTION kann eine COBOL-Zeile daher nicht mit einem Bindestrich in Spalte 7 der Folgezeile fortgesetzt werden. Es empfiehlt sich, Kommentarzeilen durch einen Stern \* in Spalte 7 zu markieren. Mit D markierte Testhilfezeilen sind erlaubt und werden vom ESQL-Precompiler ebenfalls als Kommentarzeilen interpretiert.

Die Reihenfolge der einzelnen Klauseln ist beliebig. Jede Klausel darf höchstens einmal angegeben werden.

```
EXEC SQL BEGIN DECLARE SECTION END-EXEC
```

```
EXEC SQL END DECLARE SECTION END-EXEC
```

**Anfang bzw. Abschluss einer DECLARE SECTION.** Beide müssen vollständig in jeweils einer Zeile im Bereich von Spalte 8 bis 72 stehen.

*stufennummer*

Stufennummer. Für Stufennummern gelten die COBOL-Konventionen. Die Stufennummer 66 ist nicht erlaubt. Wenn Sie mit der Precompiler-Option SOURCE-PROPERTIES den ISO-Sprachumfang festgelegt haben (siehe [Abschnitt „Eigenschaften des ESQL-COBOL-Programms festlegen“ auf Seite 98](#)), sind nur die Stufennummern 01 und 77 zulässig.

*datename*

Datenname gemäß COBOL-Regeln. Erlaubte Zeichen sind Groß- und Kleinbuchstaben, Ziffern und Bindestrich. Groß- und Kleinbuchstaben werden nicht unterschieden. *datename* darf maximal 30 Zeichen lang sein und darf kein reserviertes COBOL-Wort sein. Ebenfalls dürfen VARCHAR, TIMESTAMP, EXEC und END-EXEC nicht als *datename* verwendet werden. *datename* darf nicht mit einem Bindestrich beginnen oder enden und muss mindestens einen Buchstaben enthalten. Wenn Sie mit der Precompiler-Option SOURCE-PROPERTIES den ISO-Sprachumfang festgelegt haben (siehe [Abschnitt „Eigenschaften des ESQL-COBOL-Programms festlegen“ auf Seite 98](#)), müssen Sie einen Datennamen angeben. Namen von Benutzervariablen mit der Stufennummer 01 oder 77 müssen im Programmtext eindeutig sein. Untergeordnete Datennamen müssen innerhalb der Datengruppe eindeutig sein (siehe auch [Abschnitt „Benutzervariablen definieren“ auf Seite 35](#)). Um Namenskonflikte mit Datennamen aus dem Kommunikationsbereich zu vermeiden, sollten Sie in einer DECLARE SECTION außer den Datennamen SQLCODE und SQLSTATE keine Datennamen verwenden, die mit SQL beginnen.

*PICTURE-Klausel*

PICTURE-Klausel. Die PICTURE-Klausel für ein alphanumerisches Datenfeld muss mindestens ein X enthalten. Die Maskenzeichen A und 9 werden aus Kompatibilitätsgründen zu Vorgängerversionen von SESAM/SQL für alphanumerische Datenfelder unterstützt. Sie sollten jedoch nur das Maskenzeichen X verwenden. In der PICTURE-Klausel für ein numerisches Datenfeld müssen Sie als erstes Maskenzeichen S angeben. Bei Gleitpunktzahlen (USAGE IS COMP-1 oder USAGE IS COMP-2) ist die PICTURE-Klausel nicht zulässig (siehe [Abschnitt „Gleitpunktzahl mit einfacher Genauigkeit“ auf Seite 57](#) und [Abschnitt „Gleitpunktzahl mit doppelter Genauigkeit“ auf Seite 58](#)).

Die PICTURE-Klausel für ein nationales Datenfeld darf lediglich das Maskenzeichen N enthalten.

*USAGE-Klausel*

USAGE-Klausel. Ist keine USAGE-Klausel angegeben, wird DISPLAY angenommen. Handelt es sich aber um ein nationales Datenfeld, definiert durch das Maskenzeichen N in der PICTURE-Maske, wird USAGE NATIONAL angenommen. Die USAGE-Klausel darf nur bei Datenfeldern verwendet werden, die keine untergeordneten Datenfelder haben.

*GROUP-USAGE-Klausel*

GROUP-USAGE-Klausel. GROUP-USAGE NATIONAL legt in COBOL fest, dass eine Gruppe mit ausschließlich nationalen Unterfeldern als nationales elementares Datenfeld behandelt wird, nicht wie sonst als alphanumerisches Datenfeld.

*SIGN-Klausel*

SIGN-Klausel für numerische Datenfelder. Die SIGN-Klausel ist nur erlaubt, wenn keine USAGE-Klausel oder USAGE IS DISPLAY angegeben ist.

*VALUE-Klausel*

VALUE-Klausel. In der VALUE-Klausel werden Literale entsprechend den COBOL-Regeln angegeben. Nationale Literale werden entsprechend der COBOL-Definition durch N' oder NX' eingeleitet. Bei Angabe eines nationalen Literals in der VALUE-Klausel kann die optionale PICTURE-Angabe fehlen und wird aus dem Literal abgeleitet.

Dezimalzahlen können mit Dezimalpunkt oder mit Dezimalkomma angegeben werden. Ein mit der VALUE-Klausel angegebener alphanumerischer Wert kann abhängig von den Einstellungen des ESQL-Precompiler und des COBOL-Compiler in Hochkomma ' statt in Anführungszeichen " eingeschlossen werden.

*SYNCHRONIZED-Klausel*

Synchronized-Klausel.

*OCCURS-Klausel*

OCCURS-Klausel. Die OCCURS-Klausel darf nur verwendet werden, um die genaue Anzahl der Wiederholungen eines Datenfeldes anzugeben (Format 1, siehe COBOL2000 Handbuch „[Sprachbeschreibung](#)“ [16]). Die OCCURS-Klausel darf nicht geschachtelt werden.

ASCENDING, DESCENDING, KEY und DEPENDING werden vom Precompiler ignoriert.

*EXTERNAL-Klausel*

EXTERNAL-Klausel.

*GLOBAL-Klausel*

GLOBAL-Klausel.

## VARCHAR

Definition einer Benutzervariablen für eine Zeichenkette variabler Länge. Zu diesem Datentyp siehe [Abschnitt „Zeichenkette variabler Länge“ auf Seite 45](#).

## NVARCHAR

Definition einer Benutzervariablen für eine nationale Zeichenkette variabler Länge. Zu diesem Datentyp siehe [Abschnitt „Nationale Zeichenkette variabler Länge“ auf Seite 48](#).

## DATE

Definition einer Benutzervariablen für ein Datum. Zu diesem Datentyp siehe [Abschnitt „Datum“ auf Seite 59](#).

## TIME(3)

Definition einer Benutzervariablen für einen Zeitpunkt. Zu diesem Datentyp siehe [Abschnitt „Uhrzeit“ auf Seite 61](#).

## TIMESTAMP(3)

Definition einer Benutzervariablen für einen Zeitstempel. Zu diesem Datentyp siehe [Abschnitt „Zeitstempel“ auf Seite 63](#).

*Beispiel*

Benutzervariablen definieren:

```

DATA DIVISION.
WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC
*
*** BENUTZERVARIABLE FUER TABELLE AUFTRAG
*
01 AUFTRAG.
    02 ANR          PIC S9(9) USAGE IS BINARY.
    02 ADATUM DATE.
        03 JAHR     PIC S9(4) USAGE IS BINARY.
        03 MONAT    PIC S9(4) USAGE IS BINARY.
        03 TAG      PIC S9(4) USAGE IS BINARY.
    02 ATEXT        PIC X(30).
    02 ASTAT        PIC S9(9) USAGE IS BINARY.

    EXEC SQL END DECLARE SECTION END-EXEC
.
.
.
weitere COBOL-Datendefinitionen
.
.
.

```

## Einschränkungen bei ISO-Sprachumfang

Wenn Sie mit der Precompiler-Option SOURCE-PROPERTIES den ISO-Sprachumfang festgelegt haben (siehe [Abschnitt „Eigenschaften des ESQL-COBOL-Programms festlegen“ auf Seite 98](#)), sind die folgenden Klauseln nicht erlaubt:

- BASED
- SYNCHRONIZED
- OCCURS
- EXTERNAL
- GROUP-USAGE
- GLOBAL
- VARCHAR
- NVARCHAR
- DATE
- TIME(3)
- TIMESTAMP(3)

Für die erlaubten Klauseln gelten einige Einschränkungen. Diese sind in den Abschnitten zu den einzelnen Datentypen beschrieben (siehe [Abschnitt „Zuordnung von SQL- und COBOL-Datentypen“ auf Seite 43](#)).

## Benennung von Benutzervariablen

Der ESQL-Precompiler erkennt die Struktur geschachtelter ESQL-COBOL-Programme nicht. Daher kann es zu unerwünschten Referenzierungen kommen, wenn in einem geschachtelten ESQL-COBOL-Programm eine Benutzervariable und eine COBOL-Variable mit gleichem Namen definiert sind. Solche unerwünschten Referenzierungen können Sie durch eine systematische Benennung der Variablen vermeiden.

*Beispiel***Unerwünschte Referenzierung in einem geschachtelten ESQL-COBOL-Programm:**

```

PROGRAM-ID. A1.
DATA DIVISION.
WORKING-STORAGE SECTION.

*****
* BENUTZERVARIABLE X
*****
      EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 X PIC X(10).
      EXEC SQL END DECLARE SECTION END-EXEC.
.
.
.
PROGRAM-ID. A2.
DATA DIVISION.
WORKING-STORAGE SECTION.

*****
* COBOL-VARIABLE X
*****
01 X PIC X(11).

PROCEDURE DIVISION.
01.
      EXEC SQL
          SELECT A INTO :X  -- COBOL-VARIABLE X AUS A2 WIRD UNERWUENSCHT
                           -- REFERENZIERT
          FROM T
          END EXEC
END-PROGRAM A2.
END-PROGRAM A1.

```

## 2.3.2 Benutzervariablen in SQL-Anweisungen angeben

In einer SQL-Anweisung müssen Sie eine Benutzervariable mit einem vorangestellten Doppelpunkt kennzeichnen. Auf den Namen einer Benutzervariable darf nicht unmittelbar ein Buchstabe, eine Ziffer oder ein Bindestrich folgen.

### *Beispiel*

Benutzervariablen für Tabelle KUNDE definieren und in einer SQL-Anweisung verwenden:

```

DATA DIVISION.
WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC
*
* BENUTZERVARIABLEN FUER TABELLE KUNDE DEFINIEREN
*
    01 KUNDE.
        05 KNR          PIC S9(9).
        05 FIRMA        PIC X(40).
        05 STRASSE      PIC X(40).
    EXEC SQL END DECLARE SECTION END-EXEC
.
.
.
PROCEDURE DIVISION.
*
* BENUTZERVARIABLEN IN EINER SQL-ANWEISUNG VERWENDEN
*
    EXEC SQL
        INSERT INTO KUNDE (KNR, FIRMA, STRASSE)
        VALUES (:KNR, :FIRMA, :STRASSE)
    END EXEC

```

### 2.3.2.1 Namen untergeordneter Datenfelder qualifizieren

In einer SQL-Anweisung muss ein untergeordnetes Datenfeld so angegeben werden, dass es eindeutig identifiziert werden kann. Dazu können Sie den Namen des untergeordneten Datenfelds mit dem Namen eines übergeordneten Datenfelds qualifizieren, indem Sie den Namen des übergeordneten Datenfelds dem Namen des untergeordneten Datenfelds durch Punkt getrennt voranstellen. Datenfelder werden also im Vergleich zu COBOL in umgekehrter Reihenfolge qualifiziert.

*Beispiel*

Benutzervariable als Datengruppe definieren und verwenden:

```

DATA DIVISION.
WORKING-STORAGE SECTION.
*
* BENUTZERVARIABLEN DEFINIEREN
*
    EXEC SQL BEGIN DECLARE SECTION END-EXEC
01 ADRESSE.
    02 STRASSE    PIC X(40).
    02 PLZ        PIC S9(6).
    02 ORT        PIC X(40).
    EXEC SQL END DECLARE SECTION END-EXEC
.
.
.
PROCEDURE DIVISION.
*
* BENUTZERVARIABLE IN EINER SQL-ANWEISUNG ANSPRECHEN
*
    EXEC SQL
        SELECT STRASSE FROM KUNDE INTO :ADRESSE.STRASSE
    END-EXEC
.
.
.
*
* BENUTZERVARIABLE IN EINER COBOL-ANWEISUNG ANSPRECHEN
*
    MOVE "Edlingerstrasse" TO STRASSE OF ADRESSE.

```

**2.3.2.2 Vektoren ansprechen**

Ist eine Benutzervariable als Vektor definiert, kann ein einzelnes Element des Vektors oder ein Bereich des Vektors angesprochen werden. Um ein einzelnes Element eines Vektors anzusprechen, geben Sie den Index des Elements an. Um einen Bereich eines Vektors anzusprechen, geben Sie die Indizes für Anfang und Ende des Bereichs an. Die Zählung der Indizes von Vektoren beginnt in SQL und in COBOL mit 1.

*Beispiel*

Vektor für Farbanteile in der Tabelle FARBTAB definieren und verwenden:

```

DATA DIVISION.
WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC

```

```
*
*** VEKTOR DEFINIEREN
*
  01 FARBE.
    02 FARBANTEILE  PIC SV99 OCCURS 3 TIMES.
*
  EXEC SQL END DECLARE SECTION END-EXEC
.
.
.
PROCEDURE DIVISION.
.
.
.
*
*** VEKTOR IN SQL-ANWEISUNG VERWENDEN
*
  EXEC SQL
    SELECT RGB(1..3) INTO :FARBANTEILE(1..3)
      FROM FARBTAB
      WHERE FARBNAME = 'himmelblau'
  END-EXEC
*
.
.
.
*
*** BEREICH DES VEKTORS IN SQL-ANWEISUNG ANSPRECHEN
*
  EXEC SQL
    UPDATE FARBTAB
      SET RGB(1..2) = :FARBANTEILE(1..2)
      WHERE FARBNAME = 'himmelblau'
  END-EXEC.
*
*** EINZELNES ELEMENT DES VEKTORS ANSPRECHEN
*
  EXEC SQL
    UPDATE FARBTAB
      SET RGB(1) = :FARBANTEILE(1)
      WHERE FARBNAME = 'feuerrot'
  END-EXEC
```

### 2.3.3 Indikatorvariablen

Eine Benutzervariable kann in einer SQL-Anweisung mit einer Indikatorvariablen kombiniert werden. Eine Indikatorvariable ist eine Benutzervariable, mit der Sie die Übertragung von Werten aus der Datenbank in eine zugehörige Benutzervariable überprüfen und den NULL-Wert übertragen können.

#### 2.3.3.1 Indikatorvariable definieren

Indikatorvariablen definieren Sie wie alle Benutzervariablen innerhalb einer DECLARE SECTION in der DATA DIVISION. Der Datentyp der Indikatorvariablen muss dem SQL-Datentyp SMALLINT zugeordnet sein (siehe [Abschnitt „Kleine Ganzzahl“ auf Seite 50](#)).

Ist die Benutzervariable ein Vektor, muss die zugehörige Indikatorvariable als Vektor mit derselben Anzahl von Elementen definiert werden.

Ist der Datentyp der Benutzervariablen dem SQL-Datentyp DATE, TIME(3), TIMESTAMP(3) oder VARCHAR zugeordnet, muss die zugehörige Indikatorvariable als einfaches Datenfeld definiert werden.

#### Beispiel 1

Benutzervariablen und zugehörige Indikatorvariablen zur Tabelle AUFTRAG definieren:

```

EXEC SQL BEGIN DECLARE SECTION END-EXEC
*
**** BENUTZERVARIABLEN DEFINIEREN
*
01 FIRMA PIC X(40).
*
01 AUFTRAG.
02 ANR          PIC S9(9) USAGE IS BINARY.
02 ADATUM DATE.
03 JAHR        PIC S9(4) USAGE IS BINARY.
03 MONAT       PIC S9(4) USAGE IS BINARY.
03 TAG         PIC S9(4) USAGE IS BINARY.
02 ATEXT       PIC X(30).
02 ASTAT       PIC S9(9) USAGE IS BINARY.
*
**** ZUGEHÖRIGE INDIKATORVARIABLEN DEFINIEREN
*
01 INDFIRMA PIC S9(4) USAGE IS BINARY.
*
01 INDAUFTRAG.
02 INDANR          PIC S9(4) USAGE IS BINARY.
02 INDADATUM       PIC S9(4) USAGE IS BINARY.
02 INDATEXT        PIC S9(4) USAGE IS BINARY.
02 INDASTAT        PIC S9(4) USAGE IS BINARY.

EXEC SQL END DECLARE SECTION END-EXEC

```

*Beispiel 2*

Vektor und zugehörige Indikatorvariable zur Tabelle FARBTAB definieren:

```

EXEC SQL BEGIN DECLARE SECTION END-EXEC
*
*** BENUTZERVARIABLEN FUER FARBTAB
*
01 FARBTAB.
   02 FARBNAME    PIC X(15).
   02 RGB         PIC SV99 OCCURS 3 TIMES.
*
*** INDIKATORVARIABLEN FUER FARBTAB
*
01 INDFARBTAB.
   02 INDFARBNAME PIC S9(4) COMP.
   02 INDRGB     PIC S9(4) COMP OCCURS 3 TIMES.
*
EXEC SQL END DECLARE SECTION END-EXEC

```

### 2.3.3.2 Indikatorvariable in einer SQL-Anweisung angeben

Eine Benutzervariable darf nur mit einer Indikatorvariable kombiniert werden, wenn die Benutzervariable wie folgt verwendet wird:

- zum Abfragen von Daten aus der Datenbank
- zum Einfügen von Werten in die Datenbank
- zum Ändern von Werten in der Datenbank
- in Berechnungen (Funktionen, Ausdrücke, Prädikate, Suchbedingungen)

Indikatorvariablen müssen Sie wie alle anderen Benutzervariablen in einer SQL-Anweisung mit einem vorangestellten Doppelpunkt kennzeichnen. Sie geben die Indikatorvariable nach der Benutzervariable an, für die die Indikatorvariable gelten soll. Zwischen der Benutzervariable und der Indikatorvariable kann ein Leerzeichen stehen. Sie können die Indikatorvariable zusätzlich mit dem Schlüsselwort INDICATOR kennzeichnen.

*Beispiel*

Indikatorvariable zu einer einfachen Benutzervariable und zu einem Vektor in einer SQL-Anweisung angeben:

```

EXEC SQL
    FETCH CUR_FARBTAB
           INTO :FARBNAME    INDICATOR :INDFARBNAME,
              :RGB(1..3)    INDICATOR :INDRGB(1..3)
END-EXEC

```

### 2.3.3.3 Übertragung von Werten überprüfen

Bei der Abfrage eines Wertes aus der Datenbank mit Zuweisung an eine Benutzervariable wird die zugehörige Indikatorvariable von SESAM/SQL wie folgt belegt:

Wert	Bedeutung
0	Die Benutzervariable enthält den gelesenen Wert. Die Zuweisung war fehlerfrei.
-1	Der Wert, der zugewiesen werden sollte, ist der NULL-Wert. Die Zuweisung ist nicht erfolgt.
>0	Bei alphanumerischen Werten: Der Benutzervariable wurde eine verkürzte Zeichenkette zugewiesen. Der Wert der Indikatorvariablen gibt die Originallänge an.

Tabelle 5: In Indikatorvariablen abgelegte Werte und ihre Bedeutung

### 2.3.3.4 NULL-Wert übertragen

In Benutzervariablen ist der NULL-Wert nicht darstellbar. Deshalb sind zur Übertragung von NULL-Werten Indikatorvariablen notwendig.

Um den NULL-Wert in eine Spalte einzutragen, weisen Sie vor dem Aufruf der entsprechenden SQL-Anweisung der Indikatorvariablen einen negativen Wert zu. Bei Ausführung der SQL-Anweisung wird dann nicht der Wert der Benutzervariable verwendet, sondern der NULL-Wert.

#### *Beispiel*

Die Indikatorvariable IND1 wird mit dem Wert -2 belegt, um mit der zugehörigen Benutzervariable KNR den NULL-Wert in die Datenbank zu übertragen:

```

IF KNR = '000000000'
  THEN
    MOVE -2 TO IND1
  ELSE
    MOVE 0 TO IND1.

EXEC SQL
  UPDATE AUFTRAG
  SET KNR = :KNR INDICATOR :IND1
END-EXEC

```

## 2.4 Zuordnung von SQL- und COBOL-Datentypen

Bei der Definition einer Benutzervariablen müssen Sie einen COBOL-Datentyp wählen, dessen zugeordneter SQL-Datentyp zu der entsprechenden Spalte passt.

Eine Beschreibung der SQL-Datentypen finden Sie im Handbuch „[SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen](#)“ [2]. Eine Beschreibung der COBOL-Datentypen finden Sie im COBOL2000 Handbuch „[Sprachbeschreibung](#)“ [16].

In den folgenden Abschnitten ist für jeden SQL-Datentyp der zugeordnete COBOL-Datentyp angegeben.

### ISO-Sprachumfang

Wenn Sie mit der Precompiler-Option SOURCE-PROPERTIES den ISO-Sprachumfang festgelegt haben (siehe [Abschnitt „Eigenschaften des ESQL-COBOL-Programms festlegen“ auf Seite 98](#)), gelten bestimmte Einschränkungen bezüglich der einzelnen Klauseln, und die angegebene Reihenfolge der Klauseln muss eingehalten werden. Die Einschränkungen und die Reihenfolge werden bei den einzelnen Datentypen jeweils unter der Überschrift **ISO-Sprachumfang** beschrieben.

### Wiederholung von Maskenzeichen in der PICTURE-Klausel

Ein Maskenzeichen kann in einer PICTURE-Klausel wiederholt werden, indem es mehrfach aufgeführt wird oder ein Wiederholungsfaktor in Klammern angegeben wird. Auch Mischformen sind erlaubt.

#### *Beispiel*

Folgende Angaben sind äquivalent:

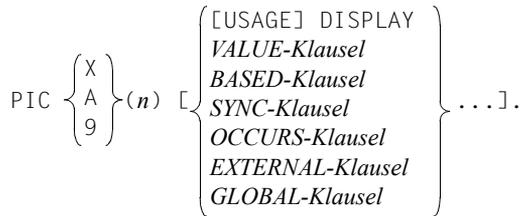
```
PIC XXX  
PIC X(3)  
PIC X(2)X
```

### Darstellung der PICTURE-, USAGE- und SIGN-Klausel

Aus Gründen der Übersichtlichkeit wird im folgenden bei der PICTURE-, USAGE- und SIGN-Klausel IS nicht angegeben. IS darf jedoch angegeben werden. Außerdem wird die PICTURE-Klausel mit PIC abgekürzt. PICTURE kann jedoch auch ausgeschrieben werden.

### 2.4.1 Zeichenkette fester Länge

Dem SQL-Datentyp **CHARACTER**(*n*) ist folgender COBOL-Datentyp zugeordnet:



#### ISO-Sprachumfang

---

PIC X(*n*) [*VALUE-Klausel*].

---

*n* Ganzzahl zwischen 1 und 256, die die Länge der Zeichenkette angibt. Bei Benutzervariablen, die nur in den SQL-Anweisungen PREPARE oder EXECUTE IMMEDIATE verwendet werden, darf die Länge der Zeichenkette bis zu 32000 Zeichen betragen. Das Maskenzeichen X muss mindestens einmal angegeben werden. Die Maskenzeichen A und 9 werden aus Kompatibilitätsgründen zu Vorgängerversionen von SESAM/SQL unterstützt. Sie sollten jedoch nur das Maskenzeichen X verwenden.

#### Beispiel

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Zeichenkette fester Länge:

SQL-Datentyp

FIRMA CHARACTER(40)

COBOL-Datentyp

01 FIRMENNAME PIC X(40).

## 2.4.2 Zeichenkette variabler Länge

Dem SQL-Datentyp **VARCHAR**( $m$ ) bzw. CHAR[ACTER] VARYING( $max$ ) ist folgender COBOL-Datentyp zugeordnet:

---

*st\_nr1* [*datename1*] VARCHAR [BASED].

$$st\_nr2\ datename2\ PIC\ S9(n)\ [USAGE]\ \left\{ \begin{array}{l} COMPUTATIONAL \\ COMPUTATIONAL-5 \\ BINARY \end{array} \right\} \left[ \left\{ \begin{array}{l} VALUE-Klausel \\ SYNC-Klausel \end{array} \right\} \dots \right].$$

$$st\_nr2\ datename3\ PIC\ \left\{ \begin{array}{l} X \\ A \\ 9 \end{array} \right\} (m)\ \left[ \left\{ \begin{array}{l} [USAGE] DISPLAY \\ VALUE-Klausel \\ SYNC-Klausel \end{array} \right\} \dots \right].$$


---

### ISO-Sprachumfang

Benutzervariablen mit Datentyp VARCHAR sind nicht erlaubt.

*st\_nr1 datename1* VARCHAR.

Vereinbart eine Datengruppe, die den SQL-Datentyp VARCHAR hat. Die Datennamen der untergeordneten Datenfelder *datename2* und *datename3* müssen unterschiedlich sein.

*st\_nr2 datename2* PIC . . .

Untergeordnetes Datenfeld mit dem Datentyp „kleine Ganzzahl“ mit  $1 \leq n \leq 4$  (siehe [Abschnitt „Kleine Ganzzahl“ auf Seite 50](#)). Der Inhalt dieses Datenfelds gibt die Länge der Zeichenkette an.

*st\_nr2 datename3* PIC . . .

Untergeordnetes Datenfeld mit dem Datentyp „Zeichenkette fester Länge“ mit  $1 \leq m \leq 32000$  (siehe [Abschnitt „Zeichenkette fester Länge“ auf Seite 44](#)). Dieses Datenfeld enthält die Zeichenkette.  $m$  ist die maximale Länge von Zeichenketten des SQL-Datentyps VARCHAR( $m$ ).

*Beispiel*

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Zeichenkette variabler Länge:

**SQL-Datentyp**

```
TITEL VARCHAR(500)
```

**COBOL-Datentyp**

```
01 TITEL VARCHAR  
02 LAENGE PIC S9(4) USAGE IS BINARY.  
02 ZEICHENKETTE PIC X(500).
```

### 2.4.3 Nationale Zeichenkette fester Länge

Dem SQL-Datentyp **NCHAR**(*n*) ist folgender COBOL-Datentyp zugeordnet:

---

```
PIC N(n) [ { [USAGE] DISPLAY
             VALUE-Klausel
             BASED-Klausel
             SYNC-Klausel
             OCCURS-Klausel
             EXTERNAL-Klausel
             GLOBAL-Klausel } ... ].
```

---

#### ISO-Sprachumfang

---

```
PIC N(n) [VALUE-Klausel].
```

---

*n* Ganzzahl zwischen 1 und 128, die die Länge der Zeichenkette (in national character positions) angibt. Das Maskenzeichen N muss mindestens einmal angegeben werden.

#### Beispiel

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine nationale Zeichenkette fester Länge:

SQL-Datentyp

```
FIRMA NCHAR(40)
```

COBOL-Datentyp

```
01 FIRMENNAME PIC N(40).
```

## 2.4.4 Nationale Zeichenkette variabler Länge

Dem SQL-Datentyp **NVARCHAR**(*m*) ist folgender COBOL-Datentyp zugeordnet:

---

*st\_nr1* [*datename1*] NVARCHAR [BASED].

$$st\_nr2\ datename2\ PIC\ S9(n)[USAGE]\ \left\{ \begin{array}{l} [COMPUTATIONAL] \\ [COMPUTATIONAL]-5 \\ [BINARY] \end{array} \right\} \left[ \left\{ \begin{array}{l} VALUE-Klausel \\ SYNC-Klausel \end{array} \right\} \dots \right].$$

$$st\_nr2\ datename3\ PIC\ N(m)\ \left[ \left\{ \begin{array}{l} [USAGE]\ DISPLAY \\ VALUE-Klausel \\ SYNC-Klausel \end{array} \right\} \dots \right].$$


---

### ISO-Sprachumfang

Benutzervariablen mit Datentyp NVARCHAR sind nicht erlaubt.

*st\_nr1 datename1* NVARCHAR.

Vereinbart eine Datengruppe, die den SQL-Datentyp NVARCHAR hat. Die Datennamen der untergeordneten Datenfelder *datename2* und *datename3* müssen unterschiedlich sein.

*st\_nr2 datename2* PIC . . .

Untergeordnetes Datenfeld mit dem Datentyp „kleine Ganzzahl“ mit  $1 \leq n \leq 4$  (siehe [Abschnitt „Kleine Ganzzahl“ auf Seite 50](#)). Der Inhalt dieses Datenfelds gibt die Länge der Zeichenkette an.

*st\_nr2 datename3* PIC . . .

Untergeordnetes Datenfeld mit dem Datentyp „Nationale Zeichenkette fester Länge“ mit  $1 \leq m \leq 16000$  (siehe [Abschnitt „Zeichenkette fester Länge“ auf Seite 44](#)). Dieses Datenfeld enthält die Zeichenkette. *m* ist die maximale Länge von Zeichenketten des SQL-Datentyps NVARCHAR(*m*).

*Beispiel*

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine nationale Zeichenkette variabler Länge:

**SQL-Datentyp**

```
TITEL NVARCHAR(500)
```

**COBOL-Datentyp**

```
01 TITEL NVARCHAR  
02 LAENGE PIC S9(4) USAGE IS BINARY.  
02 ZEICHENKETTE PIC N(500).
```

## 2.4.5 Kleine Ganzzahl

Dem SQL-Datentyp **SMALLINT** ist folgender COBOL-Datentyp zugeordnet:

---


$$\text{PIC S9}(n) \text{ [USAGE] } \left\{ \begin{array}{l} \text{COMPUTATIONAL} \\ \text{COMPUTATIONAL}-5 \\ \text{BINARY} \end{array} \right\} \left[ \begin{array}{l} \text{VALUE-Klausel} \\ \text{SYNC-Klausel} \\ \text{BASED-Klausel} \\ \text{OCCURS-Klausel} \\ \text{GLOBAL-Klausel} \\ \text{EXTERNAL-Klausel} \end{array} \right\} \dots].$$


---

### ISO-Sprachumfang

---


$$\text{PIC S9}(n) \text{ [USAGE] } \left\{ \begin{array}{l} \text{COMPUTATIONAL} \\ \text{COMPUTATIONAL}-5 \\ \text{BINARY} \end{array} \right\} \text{ [VALUE-Klausel]}.$$


---

*n* Ganzzahl zwischen 1 und 4, die die Anzahl der Stellen angibt.

Der SQL-Datentyp **SMALLINT** umfasst ganze Zahlen im Bereich von -32768 bis +32767. Der COBOL-Datentyp **PIC S9(4)** umfasst ganze Zahlen von -9999 bis +9999, also einen kleineren Wertebereich. Bei **USAGE IS BINARY** wird der Wert in COBOL-Anweisungen auf vier Dezimalstellen gekürzt. Eine COBOL-Anweisung mit einer solchen Variablen kann zu einem Überlauf führen, wenn der Wert der Variablen nicht zwischen -9999 und +9999 liegt.

*Beispiele*

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine kleine Ganzzahl für erwartete Werte zwischen -9999 und +9999:

SQL-Datentyp:

```
SEITE SMALLINT
```

COBOL-Datentyp

```
01 SEITE PIC S9(4) USAGE IS BINARY.
```

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine kleine Ganzzahl für erwartete Werte zwischen -32768 und 32767:

SQL-Datentyp:

```
SEITE SMALLINT
```

COBOL-Datentyp:

```
01 SEITE PIC S9(4) USAGE IS COMP.
```

### 2.4.6 Ganzzahl

Dem SQL-Datentyp **INTEGER** ist folgender COBOL-Datentyp zugeordnet:

---

$$\text{PIC S9}(n) \text{ [USAGE] } \left\{ \begin{array}{l} \text{COMPUTATIONAL} \\ \text{COMPUTATIONAL}-5 \\ \text{BINARY} \end{array} \right\} \left[ \begin{array}{l} \text{[VALUE-Klausel} \\ \text{SYNC-Klausel} \\ \text{BASED-Klausel} \\ \text{OCCURS-Klausel} \\ \text{GLOBAL-Klausel} \\ \text{EXTERNAL-Klausel} \end{array} \right] \dots$$

---

#### ISO-Sprachumfang

---

$$\text{PIC S9}(n) \text{ [USAGE] } \left\{ \begin{array}{l} \text{COMPUTATIONAL} \\ \text{COMPUTATIONAL}-5 \\ \text{BINARY} \end{array} \right\} \text{ [VALUE-Klausel]}.$$

---

*n* Ganzzahl zwischen 5 und 9, die die Anzahl der Stellen angibt.

Der Datentyp **INTEGER** umfasst ganze Zahlen im Bereich von -2147483648 bis +2147483647. Der COBOL-Datentyp `PIC S9(9) COMP` umfasst ganze Zahlen von -999999999 bis +999999999, also einen kleineren Wertebereich. Bei `USAGE IS BINARY` wird der Wert in COBOL-Anweisungen auf neun Dezimalstellen gekürzt. Eine COBOL-Anweisung mit einer solchen Variablen kann zu einem Überlauf führen, wenn der Wert der Variablen nicht zwischen -999999999 und +999999999 liegt.

*Beispiele*

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Ganzzahl mit erwarteten Werten zwischen -999999999 und +999999999:

SQL-Datentyp:

```
KNR INTEGER
```

COBOL-Datentyp:

```
01 KUNDENNR PIC S9(9) USAGE IS BINARY.
```

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Ganzzahl mit erwarteten Werten zwischen -2147483648 und 2147483647:

SQL-Datentyp:

```
KNR INTEGER
```

COBOL-Datentyp:

```
01 KUNDENNR PIC S9(9) USAGE IS COMP.
```

## 2.4.7 Festpunktzahl (gepackt)

Dem SQL-Datentyp **DECIMAL**( $n$ ) bzw. DECIMAL( $n,m$ ) ist folgender COBOL-Datentyp zugeordnet:

$$\left\{ \begin{array}{l} \text{PIC S9}(n) \\ \text{PIC S9}(n-m)[\text{V}[\text{9}(m)]] \\ \text{PIC SV9}(m) \end{array} \right\} [\text{USAGE}] \left\{ \begin{array}{l} \text{COMPUTATIONAL}-3 \\ \text{PACKED-DECIMAL} \end{array} \right\} \left[ \begin{array}{l} \text{VALUE-Klausel} \\ \text{BASED-Klausel} \\ \text{SYNC-Klausel} \\ \text{OCCURS-Klausel} \\ \text{GLOBAL-Klausel} \\ \text{EXTERNAL-Klausel} \end{array} \right] \dots ] .$$

### ISO-Sprachumfang

Benutzervariablen mit Datentyp DECIMAL nicht erlaubt.

- $n$  Ganzzahl zwischen 1 und 31, die die Gesamtzahl der Dezimalstellen angibt.
- $m$  Ganzzahl zwischen 1 und  $n$ , die die Anzahl der Nachkommastellen angibt.

#### Beispiele

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Festpunktzahl ohne Nachkommastellen (gepackt):

SQL-Datentyp:

```
JAHR DECIMAL(2)
```

COBOL-Datentyp:

```
01 JAHR PIC S9(2) USAGE IS PACKED-DECIMAL.
```

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Festpunktzahl mit Nachkommastellen (gepackt):

SQL-Datentyp:

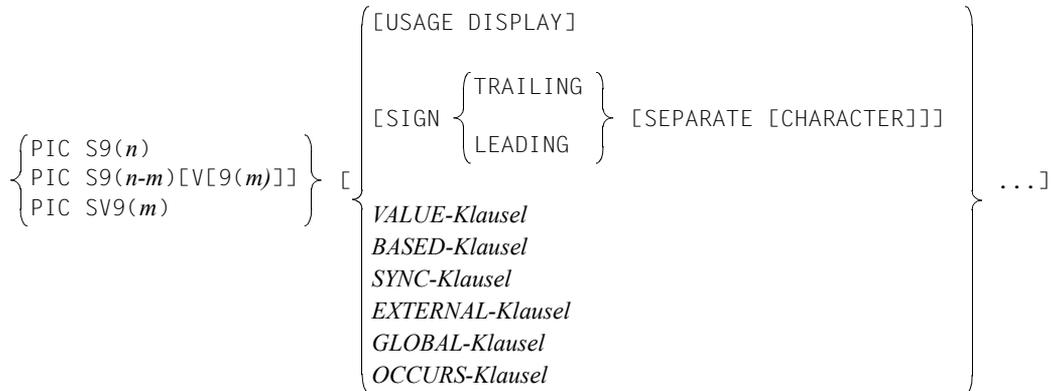
```
ZWISCHENSUM DECIMAL(6,4)
```

COBOL-Datentyp:

```
01 ZWISCHENSUM PIC S9(2)V9(4) USAGE IS PACKED-DECIMAL.
```

## 2.4.8 Festpunktzahl (ungepackt)

Dem SQL-Datentyp **NUMERIC**( $n$ ) bzw. NUMERIC( $n,m$ ) ist folgender COBOL-Datentyp zugeordnet:



### ISO-Sprachumfang

$$\left\{ \begin{array}{l} \text{PIC S9}(n) \\ \text{PIC S9}(n-m)\text{[V[9}(m)\text{]]} \\ \text{PIC SV9}(m) \end{array} \right\} \text{ [USAGE] DISPLAY LEADING SEPARATE [VALUE-Klausel]. }$$

$n$  Ganzzahl zwischen 1 und 31, die die Gesamtzahl der Dezimalstellen angibt.

$m$  Ganzzahl zwischen 1 und  $n$ , die die Anzahl der Nachkommastellen angibt.

*Beispiele*

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Festpunktzahl ohne Nachkommastellen (ungepackt):

SQL-Datentyp:

```
ANZAHL NUMERIC(8,0)
```

COBOL-Datentyp:

```
01 ANZAHL PIC S9(8)
```

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Festpunktzahl mit Nachkommastellen (ungepackt):

SQL-Datentyp:

```
PREIS NUMERIC(8,2)
```

COBOL-Datentyp:

```
01 PREIS PIC S9(6)V9(2)
```

## 2.4.9 Gleitpunktzahl mit einfacher Genauigkeit

Dem SQL-Datentyp **REAL** ist folgender COBOL-Datentyp zugeordnet:

---

```
[USAGE COMP[UTATIONAL]-1  
[BASED-Klausel]  
[VALUE-Klausel]  
[SYNC-Klausel]  
[EXTERNAL-Klausel]  
[GLOBAL-Klausel]  
[OCCURS-Klausel]
```

---

### ISO-Sprachumfang

Benutzervariablen mit Datentyp REAL nicht erlaubt.

#### *Beispiel*

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Gleitpunktzahl mit einfacher Genauigkeit:

SQL-Datentyp:

```
REALZAHL REAL
```

COBOL-Datentyp:

```
01 REALZAHL USAGE COMP-1.
```

## 2.4.10 Gleitpunktzahl mit doppelter Genauigkeit

Dem SQL-Datentyp **DOUBLE PRECISION** ist folgender COBOL-Datentyp zugeordnet:

---

```
[USAGE] COMP[UTATIONAL]-2  
[BASED-Klausel]  
[VALUE-Klausel]  
[SYNC-Klausel]  
[EXTERNAL-Klausel]  
[GLOBAL-Klausel]  
[OCCURS-Klausel]
```

---

### ISO-Sprachumfang

Benutzervariablen mit Datentyp DOUBLE PRECISION nicht erlaubt.

#### *Beispiel*

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Gleitpunktzahl mit doppelter Genauigkeit:

SQL-Datentyp:

```
ZAHL DOUBLE PRECISION
```

COBOL-Datentyp:

```
01 ZAHL USAGE COMP-2.
```

## 2.4.11 Gleitpunktzahl

Dem SQL-Datentyp **FLOAT**(*n*) ist kein COBOL-Datentyp zugeordnet.

Der Wertebereich des SQL-Datentyps **FLOAT**(*n*) ist im Wertebereich des SQL-Datentyps **DOUBLE PRECISION** bzw. bei  $n \leq 21$  auch im Wertebereich des SQL-Datentyps **REAL** enthalten (siehe [Abschnitt „Gleitpunktzahl mit doppelter Genauigkeit“ auf Seite 58](#) und [Abschnitt „Gleitpunktzahl mit einfacher Genauigkeit“ auf Seite 57](#)). Daher können Benutzervariablen mit dem Datentyp **DOUBLE PRECISION** bzw. **REAL** zum Lesen von Attributen des Datentyps **FLOAT**(*n*) verwendet werden.

## 2.4.12 Datum

Dem SQL-Datentyp **DATE** ist folgender COBOL-Datentyp zugeordnet:

---

```

st_nr1 [d_name1] DATE [BASED].
  st_nr2 d_name2 PIC S9(j) [USAGE] { COMPUTATIONAL
                                     COMPUTATIONAL]-5
                                     BINARY } [ { VALUE-Klausel
                                               SYNC-Klausel } ...].

st_nr2 d_name3 PIC S9(m) [USAGE] { COMPUTATIONAL
                                     COMPUTATIONAL]-5
                                     BINARY } [ { VALUE-Klausel
                                               SYNC-Klausel } ...].

st_nr2 d_name4 PIC S9(t) [USAGE] { COMPUTATIONAL
                                     COMPUTATIONAL]-5
                                     BINARY } [ { VALUE-Klausel
                                               SYNC-Klausel } ...].

```

---

### ISO-Sprachumfang

Benutzervariable mit Datentyp DATE nicht erlaubt.

*st\_nr1 d\_name1* DATE

Definiert eine Datengruppe, die den SQL-Datentyp DATE hat. Die Datengruppe enthält drei untergeordnete Datenfelder mit dem Datentyp „kleine Ganzzahl“.

*st\_nr2 d\_name2 ... st\_nr2 d\_name4 . . .*

Untergeordnete Datenfelder mit dem Datentyp „kleine Ganzzahl“. Das erste Datenfeld ist das Jahr, das zweite der Monat und das dritte der Tag. *j, m, t* sind Ganzzahlen zwischen 1 und 4. Die Namen der Datenfelder müssen unterschiedlich sein.

*Beispiel*

SQL-Datentyp und zugeordneter COBOL-Datentyp für ein Datum:

SQL-Datentyp:

DATUM DATE

COBOL-Datentyp:

01 DATUM DATE.

02 JAHR PIC S9(4) BINARY.

02 MONAT PIC S9(2) BINARY.

02 TAG PIC S9(2) BINARY.

## 2.4.13 Uhrzeit

Dem SQL-Datentyp **TIME(3)** ist folgender COBOL-Datentyp zugeordnet:

---

```

st_nr1 [d_name1] TIME(3) [BASED].
  st_nr2 d_name2 PIC S9(h) [USAGE] { COMPUTATIONAL
                                     COMPUTATIONAL]-5
                                     BINARY } [ VALUE-Klausel
                                               SYNC-Klausel ] ...].

st_nr2 d_name3 PIC S9(m) [USAGE] { COMPUTATIONAL
                                     COMPUTATIONAL]-5
                                     BINARY } [ VALUE-Klausel
                                               SYNC-Klausel ] ...].

st_nr2 d_name4 PIC S9(s) [USAGE] { COMPUTATIONAL
                                     COMPUTATIONAL]-5
                                     BINARY } [ VALUE-Klausel
                                               SYNC-Klausel ] ...].

st_nr2 d_name5 PIC S9(t) [USAGE] { COMPUTATIONAL
                                     COMPUTATIONAL]-5
                                     BINARY } [ VALUE-Klausel
                                               SYNC-Klausel ] ...].

```

---

### ISO-Sprachumfang

Benutzervariablen mit Datentyp TIME(3) nicht erlaubt.

*st\_nr1 d\_name1* TIME(3)

Definiert eine Datengruppe, die den SQL-Datentyp TIME(3) hat. Die Datengruppe enthält vier untergeordnete Datenfelder mit dem Datentyp „kleine Ganzzahl“.

*st\_nr2 d\_name2 ... st\_nr2 d\_name5*

Untergeordnete Datenfelder mit dem Datentyp „kleine Ganzzahl“ (siehe [Abschnitt „Kleine Ganzzahl“ auf Seite 50](#)). Das erste Datenfeld enthält die Stunden, das zweite die Minuten, das dritte die Sekunden und das vierte die Tausendstelsekunden. Die Namen der Datenfelder müssen unterschiedlich sein.

*Beispiel*

SQL-Datentyp und zugeordneter COBOL-Datentyp für eine Uhrzeit:

SQL-Datentyp:

```
ZEIT TIME(3)
```

COBOL-Datentyp:

```
01 ZEIT TIME(3).  
   02 STUNDE  PIC S9(2) BINARY.  
   02 MINUTE  PIC S9(2) BINARY.  
   02 SEKUNDE PIC S9(2) BINARY.  
   02 TAUSSEK PIC S9(3) BINARY.
```

## 2.4.14 Zeitstempel

Dem SQL-Datentyp **TIMESTAMP(3)** ist folgender COBOL-Datentyp zugeordnet:

---

```
st_nr1 [d_name1] TIMESTAMP(3).  
    st_nr2 d_name2 PIC S9(j) [VALUE-Klausel].  
    st_nr2 d_name3 PIC S9(m) [VALUE-Klausel].  
    st_nr2 d_name4 PIC S9(t) [VALUE-Klausel].  
    st_nr2 d_name5 PIC S9(h) [VALUE-Klausel].  
    st_nr2 d_name6 PIC S9(min) [VALUE-Klausel].  
    st_nr2 d_name7 PIC S9(s) [VALUE-Klausel].  
    st_nr2 d_name8 PIC S9(ts) [VALUE-Klausel].
```

---

### ISO-Sprachumfang

Benutzervariablen vom Datentyp **TIMESTAMP(3)** nicht erlaubt.

```
st_nr1 d_name1 TIMESTAMP(3).
```

Definiert eine Datengruppe, die den SQL-Datentyp **TIMESTAMP(3)** hat. Die Datengruppe enthält sieben untergeordnete Datenfelder mit dem Datentyp „kleine Ganzzahl“.

```
st_nr2 d_name2 ... st_nr2 d_name8 ...
```

Unabhängige Datenfelder mit dem Datentyp „kleine Ganzzahl“ (siehe [Abschnitt „Kleine Ganzzahl“ auf Seite 50](#)). Die Datenfelder enthalten das Jahr, den Monat, den Tag, die Stunde, die Minute, die Sekunde bzw. die Tausendstelsekunden. Die Namen der Datenfelder müssen unterschiedlich sein.

*Beispiel*

SQL-Datentyp und zugeordneter COBOL-Datentyp für einen Zeitstempel:

SQL-Datentyp:

```
ZEITSTEMPEL TIMESTAMP(3)
```

COBOL-Datentyp:

```
01 ZEITSTEMPEL TIMESTAMP(3).  
  02 JAHR      PIC S9(4)  BINARY.  
  02 MONAT    PIC S9(2)  BINARY.  
  02 TAG      PIC S9(2)  BINARY.  
  02 STUNDE   PIC S9(2)  BINARY.  
  02 MINUTE   PIC S9(2)  BINARY.  
  02 SEKUNDE  PIC S9(2)  BINARY.  
  02 TAUSSEK  PIC S9(3)  BINARY.
```

## 2.4.15 Vektoren

Für multiple Spalten mit mehreren Ausprägungen können Sie mit der OCCURS-Klausel Vektoren definieren. Es dürfen nur die Stufennummern 02 bis 49 verwendet werden.

### ISO-Sprachumfang

OCCURS-Klausel bei Benutzervariablen nicht erlaubt.

OCCURS-Klauseln dürfen in einer DECLARE SECTION nicht geschachtelt werden.

#### *Beispiel*

Multiple Spalte mit drei Ausprägungen und zugeordneter COBOL-Datentyp:

SQL-Datentyp:

```
RGB(3)
```

```
NUMERIC(2,2)
```

COBOL-Datentyp:

```
01 RGB.
```

```
02 FARBANTEILE PIC SV99 OCCURS 3 TIMES.
```

## 2.5 SQL-Anweisungen in einem ESQL-COBOL-Programm

Eine SQL-Anweisung muss mit EXEC SQL eingeleitet und mit END-EXEC abgeschlossen werden. Die SQL-Anweisungen WHENEVER, DECLARE CURSOR, CREATE TEMPORARY VIEW und INCLUDE können an beliebiger Stelle im Programmtext eingebettet werden. Alle anderen SQL-Anweisungen müssen in einer PROCEDURE DIVISION von ESQL-COBOL-Programmen eingebettet werden.

Bei der Vorübersetzung werden die SQL-Anweisungen WHENEVER, DECLARE CURSOR, CREATE TEMPORARY VIEW und INCLUDE vom ESQL-Precompiler entfernt oder durch Kommentarzeilen ersetzt. Alle anderen eingebetteten SQL-Anweisungen ersetzt der ESQL-Precompiler durch COBOL-Anweisungen. Mit der Precompiler-Option PRECOMPILER-ACTION, Parameter ESQL-STATEMENTS, können Sie festlegen, ob die ursprünglichen SQL-Anweisungen und die Zeichenfolgen EXEC SQL und END-EXEC aus dem Programmtext entfernt werden oder als Kommentare erhalten bleiben. Zeilen, die nicht zwischen EXEC SQL und END-EXEC eingebettet sind, werden in das vom ESQL-Precompiler erzeugte COBOL-Programm unverändert übernommen. Steht hinter dem END-EXEC ein Punkt, so wird dieser Punkt nach den eingefügten COBOL-Anweisungen in den Programmtext übernommen. Der so entstandene Programmtext muss in COBOL erlaubt sein.

---

EXEC SQL

*SQL-Anweisung*

END-EXEC

---

Die Zeichenfolgen EXEC SQL, END-EXEC und die eingebettete SQL-Anweisung müssen in den Spalten 8-72 des COBOL-Programmtextes stehen. Die Spalte 7 muss ein Leerzeichen enthalten. Die Spalten 1-6 und 73-80 können wie in COBOL verwendet werden. Es dürfen keine Fortsetzungszeilen mit Bindestrich – in Spalte 7 innerhalb der SQL-Anweisung stehen. COBOL-Kommentarzeilen dürfen zwischen EXEC SQL und END-EXEC eingebettet werden.

EXEC SQL

Zusammengehörende Zeichenfolge, die vollständig in einer Zeile stehen muss.

*SQL-Anweisung*

Zwischen EXEC SQL und END-EXEC darf jeweils nur eine SQL-Anweisung stehen. Innerhalb der SQL-Anweisung dürfen beliebig viele Leerzeichen zur Trennung der Anweisungsteile verwendet werden.

Temporary Views und Cursor müssen im Text des ESQL-COBOL-Programms vor ihrer ersten Verwendung definiert sein. Benutzervariablen müssen im Text des ESQL-COBOL-Programms vor ihrer Verwendung in SQL-Anweisungen definiert sein. Bei einer Benutzervariablen, die in einer DECLARE CURSOR-Anweisung verwendet wird, muss die Definition der Benutzervariable bei jeder OPEN CURSOR-Anweisung für diesen Cursor sichtbar sein.

END-EXEC

Zusammengehörende Zeichenfolge, die vollständig in einer Zeile stehen muss. Der Rest der Zeile hinter END-EXEC wird unverändert in das COBOL-Programm übernommen. Er muss daher in COBOL an dieser Stelle erlaubt sein.

### *Beispiel 1*

UPDATE-Anweisung in ein ESQL-COBOL-Programm einbetten:

```
EXEC SQL
  UPDATE KONTAKT
    SET ABTEILUNG = :ABTEILUNG
    WHERE KONR = '11'
END-EXEC
```

### *Beispiel 2*

WHENEVER-Anweisung einbetten:

```
PARAGRAPH01.
  EXEC SQL
    WHENEVER SQLERROR GOTO PARAGRAPH99
  END-EXEC
PARAGRAPH02.
```

### Erläuterung

Bei der Vorübersetzung werden die Zeichenfolgen EXEC SQL und END-EXEC sowie die WHENEVER-Anweisung entfernt. Daher darf hinter END-EXEC kein Punkt stehen, da folgender Programmtext in COBOL nicht erlaubt ist:

```
PARAGRAPH01.
.
PARAGRAPH02.
```

## 2.6 SQL-Kommentare in einem ESQL-COBOL-Programm

In SQL-Anweisungen können Sie zur Dokumentation des Programms SQL-Kommentare einfügen.

Ein SQL-Kommentar beginnt mit der Zeichenfolge `--` und endet mit dem Ende der Zeile. Er muss innerhalb von Spalte 8 - 72 stehen.

Die Zeichenfolge `--` leitet keinen Kommentar ein, wenn sie in einem alphanumerischen Literal, einem Namen in Anführungszeichen oder einem Datennamen auftritt. Alphanumerische Literale und Namen in Anführungszeichen beschreibt das Handbuch „[SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen](#)“ [2].

### *Beispiel*

SQL-Kommentare angeben:

```
EXEC SQL
-- ABTEILUNG WIRD AKTUALISIERT
  UPDATE KONTAKT
    SET ABTEILUNG = :ABTEILUNG
    WHERE KONR = '11'  -- HAT ABTEILUNG GEWECHSELT
END-EXEC
```

## 2.7 Kommunikationsbereich

Jedes ESQL-COBOL-Programm muss den Kommunikationsbereich enthalten. Im Kommunikationsbereich werden Informationen zu ausgeführten SQL-Anweisungen hinterlegt. Die Informationen können im ESQL-COBOL-Programm abgefragt werden und sind für die Fehlerbehandlung und Erfolgskontrolle von Bedeutung (siehe [Abschnitt „Fehlerbehandlung und Erfolgskontrolle“ auf Seite 73](#)). ESQL-COBOL-Anwendungen sollten die Werte der Variablen im Kommunikationsbereich nicht verändern.

### 2.7.1 Aufbau des Kommunikationsbereichs

Für ESQL-COBOL gibt es zwei Varianten des Kommunikationsbereichs. Diese unterscheiden sich nur durch das Datenfeld SQLCODE. Im Datenfeld SQLCODE wird der Rückgabewert von SQL-Anweisungen abgelegt. Zukünftige Versionen der SQL-Norm unterstützen das Datenfeld SQLCODE nicht mehr. Rückgabewerte von SQL-Anweisungen werden zukünftig nur noch im Datenfeld SQLSTATE abgelegt.

#### **Variante ohne Datenfeld SQLCODE**

Das Datenfeld SQLCODE ist nicht im Kommunikationsbereich enthalten. Für ESQL-COBOL-Anwendungen, die neu für SESAM/SQL entwickelt werden, sollten Sie diese Variante verwenden.

#### **Variante für Vorgängerversionen von ESQL-COBOL**

In dieser Variante ist das Datenfeld SQLCODE im Kommunikationsbereich enthalten. Diese Variante können Sie für Anwendungen verwenden, die für Vorgängerversionen von SESAM/SQL entwickelt wurden.

**Variante ohne SQLCODE**

```

01  SQLca.
    05  SQLstatementid  PIC  S9(9) BINARY.
    05  SQLcallcount   PIC  S9(9) BINARY.
    05  SQLidmark      PIC  X(16).
    05  SQLline        PIC  S9(9) BINARY.

01  SQLda.
    05  SQLda01        PIC  S9(4) BINARY.
       88  SQLda01val  VALUE mmm.
    05  SQLda02        PIC  S9(4) BINARY.
    05  SQLda03        PIC  S9(4) BINARY.
    05  SQLda04        PIC  S9(4) BINARY.
    05  SQLerrline     PIC  S9(4) BINARY.
    05  SQLerrcol      PIC  S9(4) BINARY.
    05  SQLda07        PIC  S9(4) BINARY.
    05  SQLda08        PIC  X(5).
    05  SQLCLI-SQLSTATE redefines SQLda08 PIC X(5)
    05  SQLerrm        PIC  X(240).
    05  SQLda10        PIC  X.
    05  SQLda21        PIC  S(9)  BINARY.
    05  SQLda22        PIC  X(4).
    05  SQLda23        PIC  9(4)  BINARY.
    05  SQLda24        PIC  X(2).
    05  SQLrowcount    PIC  S9(9) BINARY.
    05  SQLda99        PIC  X(nnn).

```

**Variante für Vorgängerversionen von ESQL-COBOL**

```

01  SQLca.
    05  SQLstatementid  PIC  S9(9) BINARY.
    05  SQLcallcount   PIC  S9(9) BINARY.
    05  SQLidmark      PIC  X(16).
    05  SQLline        PIC  S9(9) BINARY.
    05  SQLCODE        PIC  S9(4) BINARY.

```

```

01  SQLda.
    .
    .      (Aufbau wie in Variante 1)
    .

```

SQLca

Datengruppe, die Informationen zu ausgeführten SQL-Anweisungen enthält.

SQLstatementid

SQLcallcount

SQLidmark

für interne Zwecke reserviert.

SQLline

enthält die Zeilennummer des vom ESQL-Precompiler erzeugten COBOL-Programms, in der die zuletzt ausgeführte SQL-Anweisung beginnt.

SQLCODE

enthält den Rückgabewert der zuletzt ausgeführten SQL-Anweisung. Eine Übersicht über die Rückgabewerte finden Sie im [Abschnitt „Fehlerbehandlung und Erfolgskontrolle“ auf Seite 73](#). Die Beschreibung einzelner Rückgabewerte und ihrer Bedeutung finden Sie im Handbuch „[Meldungen](#)“ [7].

Dieses Datenfeld ist nur in der Variante des Kommunikationsbereichs enthalten, die kompatibel ist zu Vorgängerversionen von SESAM/SQL.

SQLda

Diagnosebereich für SESAM/SQL.

SQLda01, SQLda02, ... SQLda99

für interne Zwecke reserviert. In zukünftigen SESAM/SQL-Versionen können sich die Namen dieser Datenfelder und deren Reihenfolge ändern. Daher sollten Sie sich beim Entwickeln von Anwendungen nicht auf die Reihenfolge dieser Datenfelder verlassen und ihren Inhalt nicht im ESQL-COBOL-Programm interpretieren.

SQLerrline

enthält im Fehlerfall die Zeilennummer der Stelle im Text einer vorbereiteten Anweisung, in der ein Fehler aufgetreten ist. Enthält 0 (Null), wenn die Stelle nicht bestimmt werden konnte oder nicht sinnvoll ist.

SQLerrcol

enthält im Fehlerfall die Spaltennummer der Stelle im Text einer vorbereiteten Anweisung, in der ein Fehler aufgetreten ist. Enthält 0 (Null), wenn die Stelle nicht bestimmt werden konnte oder nicht sinnvoll ist.

SQLerm

enthält nach Ausführung einer SQL-Anweisung, die in SQLSTATE einen anderen Rückgabewert als 00000 liefert, einen Meldungstext. Der Text enthält die Fehlerklasse (W für WARNING oder E für ERROR), die Meldungsnummer SQLnnnn und den Meldungstext.

SQLrowcount

enthält nach Ausführung der entsprechenden Anweisungen folgende Information:

- nach einer INSERT-Anweisung die Anzahl der eingefügten Sätze
- nach einer UPDATE- oder DELETE-Anweisung mit einer Suchbedingung die Anzahl der Sätze, für die die Suchbedingung erfüllt ist
- nach einer UPDATE- oder DELETE-Anweisung ohne WHERE-Klausel die Anzahl der Sätze in der referenzierten Tabelle
- nach einer UNLOAD-Anweisung die Anzahl der ausgegebenen Sätze
- nach einer LOAD-Anweisung die Anzahl der zugeladenen Sätze
- nach einer EXPORT-Anweisung die Anzahl der in die Export-Datei kopierten Sätze
- nach einer IMPORT-Anweisung die Anzahl der aus der Export-Datei kopierten Sätze.

Sonst ist der Inhalt nicht definiert.

## 2.7.2 Kommunikationsbereich bereitstellen

Der Kommunikationsbereich muss in die DATA DIVISION des ESQL-COBOL-Programms eingefügt werden. Dazu verwenden Sie die SQL-INCLUDE-Anweisung oder die COBOL-COPY-Anweisung. Wenn Sie die SQL-INCLUDE-Anweisung verwenden, müssen Sie die Bibliothek mit dem Kommunikationsbereich bei der Vorübersetzung als INCLUDE-Bibliothek zuweisen (siehe [Abschnitt „INCLUDE-Elemente“ auf Seite 77](#)). Verwenden Sie die COBOL-COPY-Anweisung, müssen Sie die Bibliothek mit dem Kommunikationsbereich bei der Übersetzung mit dem Linknamen COBLIB als COBOL-COPY-Bibliothek zuweisen.

Folgende Bibliotheken enthalten den Kommunikationsbereich:

- SYSLIB.ESQL-COBOL.030.INCL-V2  
enthält den Kommunikationsbereich ohne SQLCODE für ESQL-COBOL-Anwendungen für SESAM/SQL

### ESQL-COBOL-Anwendungen für SESAM/SQL

Für ESQL-COBOL-Anwendungen für SESAM/SQL sollten Sie den Kommunikationsbereich ohne SQLCODE bereitstellen. Zusätzlich müssen Sie in einer DECLARE SECTION das Datenfeld SQLSTATE bzw. SQLCODE definieren. Es empfiehlt sich, SQLSTATE zu verwenden.

### SQLSTATE bzw. SQLCODE definieren

SQLSTATE bzw. SQLCODE muss im Programmtext vor der ersten SQL-Anweisung definiert werden. SQLSTATE muss innerhalb einer DECLARE SECTION definiert werden. SQLCODE kann innerhalb oder außerhalb einer DECLARE SECTION definiert werden.

Sind in einem Programm sowohl SQLSTATE als auch SQLCODE definiert, wird das Feld SQLCODE nicht mehr versorgt. SQLCODE enthält immer den Wert 0. Eine Programmsteuerung, die auf SQLCODE aufbaut, funktioniert dann nicht mehr.

Der Datentyp von SQLSTATE muss dem SQL-Datentyp CHAR(5) entsprechen. Der Datentyp von SQLCODE muss dem SQL-Datentyp SMALLINT entsprechen. Bei der Definition von SQLSTATE bzw. SQLCODE darf die OCCURS-Klausel nicht angegeben werden, auch nicht in einem übergeordneten Datenfeld.

### Beispiel

SQLSTATE definieren und Kommunikationsbereich ohne SQLCODE einfügen. Voraussetzung ist die Zuweisung der Bibliothek SYSLIB.ESQL-COBOL.030.INCL-V2 als INCLUDE-Bibliothek.

```
DATA DIVISION. WORKING-STORAGE SECTION.
*****
* SQLSTATE DEFINIEREN
*****
EXEC SQL BEGIN DECLARE SECTION END-EXEC
01 SQLSTATE PIC X(5).

.
.
EXEC SQL END DECLARE SECTION END-EXEC
*****
* KOMMUNIKATIONSBEREICH EINFUEGEN
*****
EXEC SQL
    INCLUDE SQLCA
END-EXEC

.
.
.
```

## 2.7.3 Fehlerbehandlung und Erfolgskontrolle

Um eine ordnungsgemäße Datenbankbearbeitung zu gewährleisten, müssen Sie nach jeder ausführbaren SQL-Anweisung prüfen, ob die SQL-Anweisung erfolgreich ausgeführt wurde. Dazu fragen Sie den Wert des Datenfelds SQLSTATE bzw. SQLCODE ab. SQLSTATE bzw. SQLCODE enthält nach jeder ausführbaren SQL-Anweisung den Rückgabewert der Anweisung. Die Beschreibung der einzelnen Rückgabewerte finden Sie im Handbuch „Meldungen“ [7]. Weitere Informationen zur Fehlerbehandlung und Erfolgskontrolle erhalten Sie durch Abfragen des Kommunikationsbereichs, siehe [Abschnitt „Kommunikationsbereich“ auf Seite 69](#).

Die folgende Tabelle gibt einen Überblick über die Rückgabewerte in SQLSTATE bzw. SQLCODE und deren Bedeutung.

SQLSTATE	SQLCODE	Bedeutung
00000	0	Die SQL-Anweisung wurde erfolgreich ausgeführt.
01 $nnn$	$0 < nnn < 100$	Warnung. Die SQL-Anweisung wurde ausgeführt.
02 $nnn$	100	Keine Daten gelesen, z.B. Tabellenende erreicht oder Tabelle leer.
restliche Werte	$< 0$	Fehler. Die SQL-Anweisung wurde nichtausgeführt.
40 $nnn$	$< -1000$	Fehler. Die Transaktion wurde zurückgesetzt.

Tabelle 6: Überblick über die Rückgabewerte in SQLSTATE und SQLCODE

Sie haben zwei Möglichkeiten, den Programmablauf abhängig vom Rückgabewert der SQL-Anweisung zu steuern:

- mit COBOL-Anweisungen
- mit der SQL-Anweisung WHENEVER

### 2.7.3.1 Programmablauf mit COBOL-Anweisungen steuern

Um den Rückgabewert in SQLSTATE bzw. SQLCODE abzufragen und auf eventuelle Fehler zu reagieren, können Sie COBOL-Anweisungen verwenden und eigene Fehlerbehandlungsroutinen entwickeln.

*Beispiel*

SQLSTATE abfragen und entsprechend verzweigen:

```

ANALYSE-SQL-FEHLER.
  EVALUATE SQLSTATE(1:2) ALSO SQLSTATE(3:3)
    WHEN "00" THROUGH "01" ALSO ANY
      CONTINUE
    WHEN "02"           ALSO "000"
      PERFORM CURSOR-AM-ENDE
    .
    .
    .
  END-EVALUATE.

```

### 2.7.3.2 Programmablauf mit der SQL-Anweisung WHENEVER steuern

Mit der SQL-Anweisung WHENEVER legen Sie die Aktionen fest, die ausgeführt werden, wenn das Tabellenende erreicht bzw. die Tabelle leer ist oder ein Fehler aufgetreten ist. Mit der WHENEVER-Anweisung können Sie nicht differenziert auf einzelne Fehlerfälle eingehen.

---

WHENEVER	{	NOT FOUND	}	{	GO TO [ : ]name	}
		SQLERROR			GOTO [ : ]name	
					CONTINUE	

---

NOT FOUND

Bedingung Tabellenende erreicht oder Tabelle leer.

SQLERROR

Fehlerfall. Eine differenzierte Unterscheidung zwischen einzelnen Fehlerfällen ist hier nicht möglich.

GO TO [ : ]name oder GOTO [ : ]name

Gibt an, wo das Programm fortgesetzt wird. *name* ist ein Kapitelname oder ein Paragraphenname. Der Paragraphenname darf nicht qualifiziert werden. Das angegebene Sprungziel muss nach den COBOL-Konventionen gültig sein. Der Doppelpunkt wird aus Kompatibilitätsgründen zu Vorgängerversionen von SESAM/SQL unterstützt. In neuen Anwendungen sollte der Doppelpunkt nicht verwendet werden.

CONTINUE

Entwertet die Ausnahmebehandlung. Es erfolgt keine Verzweigung.

Die WHENEVER-Anweisung bleibt gültig, bis eine der folgenden Bedingungen eintritt:

- Es folgt im Programmtext eine WHENEVER-Anweisung für dieselbe Fehlerklasse (NOT FOUND oder SQLERROR).
- Es folgt im Programmtext eine Entwertung für diese Fehlerklasse mit CONTINUE.

*Beispiel***Ausnahmebehandlung angeben und entwerten:**

```
*****
* AUSNAHMEBEHANDLUNG ANGEBEN
*****
EXEC SQL
    WHENEVER SQLERROR GOTO MELDEN
END-EXEC
.
.
.
*****
* AUSNAHMEBEHANDLUNG ENTWERTEN
*****
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC
.
.
.
MELDEN SECTION.
*****
* IM FEHLERFALL: MELDUNG AUSGEBEN UND TRANSAKTION ZURUECKSETZEN
*****
DISPLAY "FEHLER !" UPON DSG

EXEC SQL
    ROLLBACK WORK
END-EXEC
```

## 2.8 INCLUDE-Elemente

Mehrfach verwendete Programmteile können Sie als INCLUDE-Elemente in einer PLAM-Bibliothek abspeichern und bei Bedarf mit der SQL-Anweisung INCLUDE in ein ESQL-COBOL-Programm aufnehmen. Während der Vorübersetzung ersetzt der ESQL-Precompiler die SQL-Anweisung INCLUDE sowie die Zeichenfolgen EXEC SQL und END-EXEC durch den Text des INCLUDE-Elements.

Ein INCLUDE-Element darf ESQL-COBOL-Text oder COBOL-Text enthalten. INCLUDE-Elemente dürfen auch die SQL-Anweisung INCLUDE enthalten (Schachtelung). Die maximale Schachtelungstiefe beträgt 9.

Die SQL-Anweisung INCLUDE ist eine Erweiterung zur SQL-Norm. Wenn Sie in der Precompiler-Option SOURCE-PROPERTIES den ISO-Sprachumfang festlegen (siehe [Abschnitt „Eigenschaften des ESQL-COBOL-Programms festlegen“ auf Seite 98](#)), ist die SQL-Anweisung INCLUDE nicht erlaubt.

Mit der Precompiler-Option INCLUDE-LIBRARY geben Sie die Bibliotheken an, aus denen INCLUDE-Elemente eingefügt werden können (siehe [Abschnitt „INCLUDE-Bibliotheken angeben“ auf Seite 89](#)).

*Beispiel*

SQL-INCLUDE-Bibliothek ESQL.INCLUDE.LIB.1 zuweisen und SQL-INCLUDE-Element ERRCOP einfügen:

1. SQL-INCLUDE-Bibliothek über Precompiler-Option zuweisen:

```
//PRECOMPILE      -  
//INCLUDE-LIBRARY = ESQL.INCLUDE.LIB.1  -  
.  
.  
  weitere Precompiler-Optionen  
.  
.  
//END
```

2. SQL-INCLUDE-Element mit einer SQL-INCLUDE-Anweisung aus der SQL-INCLUDE-Bibliothek in ein ESQL-COBOL-Programm einfügen:

```
DATA DIVISION.  
WORKING-STORAGE SECTION.  
.  
.  
  EXEC SQL  
    INCLUDE ERRCOP  
  END-EXEC  
.  
.
```

**Erläuterung:** Bei der Vorübersetzung wird der Programmtext

```
EXEC SQL  
  INCLUDE ERRCOP  
END-EXEC
```

durch den Inhalt des SQL-INCLUDE-Elements ERRCOP ersetzt.

---

## 3 ESQL-COBOL-Programm vorübersetzen

Dieses Kapitel behandelt folgende Themen:

- ESQL-Precompiler aufrufen und steuern
- ESQL-Precompiler-Optionen
- Beendungsverhalten des ESQL-Precompiler und Fehlermeldungen

## 3.1 ESQL-Precompiler aufrufen und steuern

Sie können den ESQL-Precompiler im Dialog oder in einer Prozedur aufrufen. Die Optionen zur Steuerung des ESQL-Precompiler können Sie beim Aufruf des ESQL-Precompiler im Dialog bzw. in der Prozedur mit angeben. Die Precompiler-Option SOURCE-PROPERTIES zum Festlegen von Eigenschaften des ESQL-COBOL-Programms können Sie auch direkt im ESQL-COBOL-Programm angeben.

### Precompiler-Option SOURCE-PROPERTIES im ESQL-COBOL-Programm angeben

Mit folgender Anweisung können Sie die Precompiler-Option SOURCE-PROPERTIES im ESQL-COBOL-Programm angeben:

```
//PRECOMPILE SOURCE-PROPERTIES=parameter
//END
```

Sie dürfen nur eine PRECOMPILE-Anweisung angeben. Diese muss in der ersten Zeile und ersten Spalte des ESQL-COBOL-Programms anfangen.

Die Angabe der Precompiler-Option im ESQL-COBOL-Programm hat Vorrang vor einer eventuell vorhandenen weiteren Angabe (z.B. Angabe der Precompiler-Option beim Aufruf des Precompiler im Dialog). Eine Ausnahme bildet der Parameter REPLACE-TOKENS. Bei diesem Parameter werden alle Angaben ausgewertet. Eine ausführliche Beschreibung der Precompiler-Option SOURCE-PROPERTIES finden Sie im [Abschnitt „Eigenschaften des ESQL-COBOL-Programms festlegen“ auf Seite 98](#).

### 3.1.1 Benötigte Bibliotheken und Dateien zuweisen

Bevor Sie den ESQL-Precompiler starten, müssen Sie folgende Bibliotheken zuweisen:

- SESAM/SQL-Modulbibliothek
- CRTE-Bibliothek

#### SESAM/SQL Modulbibliothek zuweisen

Sie können eine der folgenden Zuweisungen verwenden:

```
/SET-FILE-LINK LINK-NAME=SESAMOML , FILE-NAME=sesam-modlib (1)
/SET-TASKLIB LIBRARY=sesam-modlib (2)
/SET-FILE-LINK LINK-NAME=BLSLIBnn , FILE-NAME=sesam-modlib (3)
```

- (1) SESAM-Modulbibliothek mit dem Linknamen SESAMOML zuweisen.
- (2) SESAM-Modulbibliothek als TASKLIB zuweisen.
- (3) SESAM-Modulbibliothek als BLSLIB zuweisen mit  $00 \leq nn \leq 99$

### CRTE-Bibliothek zuweisen

Die CRTE-Bibliothek muss als BLSLIB zugewiesen werden. Es empfiehlt, sich BLSLIB00 zu verwenden:

```
/SET FILE=LINK LINK=NAME=BLSLIB00,FILE=NAME=crite-lib
```

## 3.1.2 Vorübersetzung mit Datenbankkontakt

Bei einer Vorübersetzung mit Datenbankkontakt weisen Sie üblicherweise zusätzlich eine Konfigurationsdatei zu mit CONNECT-SESAM-CONFIGURATION oder über den Linknamen SESCONF. Wenn Sie keine Konfigurationsdatei zuweisen, gelten die Standardwerte. Eine ausführliche Beschreibung der Konfigurationsdatei für den independent und den linked-in DBH finden Sie im „[Basishandbuch](#)“ [1].

## 3.1.3 ESQL-Precompiler starten

Sie können den ESQL-Precompiler mit dem Kommando START-ESQLCOB oder dem Kommando START-PROGRAM starten. Wenn Sie den ESQL-Precompiler mit einer Jobvariable überwachen möchten, haben Sie folgende Möglichkeiten:

- Sie verwenden das Kommando START-ESQLCOB und geben die Precompiler-Option MONJV an (siehe [Abschnitt „Jobvariable angeben“ auf Seite 92](#)).
- Sie verwenden das Kommando START-PROGRAM und geben beim Aufruf den Parameter MONJV an.

### ESQL-Precompiler mit START-ESQLCOB starten

Sie können den ESQL-Precompiler mit dem Kommando START-ESQLCOB wie folgt starten:

```
START-ESQLCOB precompiler-option , . . .
```

```
START-ESQLCOB
  Aufruf des ESQL-Precompiler
```

*precompiler-option*

ESQL-Precompiler-Option zum Steuern der Vorübersetzung. Die Beschreibung der einzelnen ESQL-Precompiler-Optionen finden Sie im [Abschnitt „ESQL-Precompiler-Optionen“ auf Seite 84](#).

## ESQL-Precompiler mit START-PROGRAM starten

Eine ausführliche Beschreibung des Kommandos START-PROGRAM finden Sie im Handbuch „[BS2000/OSD-BC Kommandos Band 1 - 5](#)“ [18]. Sie können den ESQL-Precompiler mit dem START-PROGRAM-Kommando wie folgt starten:

```
//START-PROGRAM FROM-FILE=*MODULE(LIBRARY=precompiler_bibl,ELEMENT=ESQLCOB - (1)
/                                     ,PROGRAM-MODE=ANY - (2)
/                                     ,RUN-MODE=ADVANCED - (3)
/                                     (ALTERNATE-LIBRARIES=YES -
/                                     ,LOAD-INFORMATION=REFERENCES -
/                                     ,UNRESOLVED-EXTRNS=DELAY) -
/                                     )
//PRECOMPILE precompiler-option, ... (4)
//END (5)
```

- (1) Aufruf des ESQL-Precompiler *precompiler\_bibl* ist die Bibliothek, die den Precompiler enthält. Der Standardname der Bibliothek ist SYSLNK.ESQL-COBOL.030.
- (2) Die Module der Ladeinheit können oberhalb oder unterhalb 16 Mbyte geladen werden.
- (3) Der DBL, der durch START-PROGRAM implizit aufgerufen wird, arbeitet in einem Betriebsmodus, der neue Funktionen (ab BS2000 V10.0A) unterstützt. Diese Angabe ist notwendig, damit Bindelademodule verarbeitet werden können. Alternative Bibliotheken werden durchsucht. Nicht aufgelöste Externverweise werden zu einem späteren Zeitpunkt aufgelöst.
- (4) Leitet die Angabe der Precompiler-Optionen zum Steuern der Vorübersetzung ein. Die Precompiler-Option MONJV ist hier nicht erlaubt. Wenn Sie den ESQL-Precompiler mit Jobvariablen überwachen möchten, müssen Sie im Kommando START-PROGRAM den Parameter MONJV angeben. Eine ausführliche Beschreibung der einzelnen ESQL-Precompiler-Optionen finden Sie im [Abschnitt „ESQL-Precompiler-Optionen“ auf Seite 84](#).
- (5) Beendet die Angabe der ESQL-Precompiler-Optionen.

**Kommandofolge zum Vorübersetzen eines ESQL-COBOL-Programms**

```

/SET-FILE-LINK LINK-NAME=BLSLIBnn , FILE=crte-lib (1)
/SET-FILE-LINK LINK-NAME=SESAMOML , FILE-NAME=sesam-modlib (2)
/CONNECT-SESAM-CONFIGURATION TO-FILE=globale konfigurationsdatei , -
/ CONFIGURATION-LINK=linkname
oder
/SET-FILE-LINK LINK-NAME=SESCONF , FILE-NAME=dateiname (3)
/START-ESQLCOB - (4)
/ SOURCE=esql-cobol-programm - (5)
/ , INCLUDE-LIBRARY=SYSLIB.ESQL-COBOL.030.INCL-V2 - (6)
/ , PRECOMPILER-ACTION=(SQL-ENTRY-NAME=einsprungsname) - (7)
/ , SOURCE-PROPERTIES=(CATALOG=datenbankname -
/ , SCHEMA=schemaname -
/ , AUTHORIZATION=' berechtigungsschlüssel ' - (8)
/ )

```

- (1) CRTE-Bibliothek als BLSLIB zuweisen. Es empfiehlt sich, BLSLIB00 zu verwenden.
- (2) SESAM/SQL-Modulbibliothek zuweisen.
- (3) Konfigurationsdatei zuweisen. Diese Zuweisung ist nur erforderlich, wenn die Vorübersetzung mit Datenbankkontakt durchgeführt werden soll (siehe [Abschnitt „Vorübersetzung steuern“ auf Seite 93](#)).
- (4) ESQL-Precompiler aufrufen.
- (5) ESQL-COBOL-Programm, das vorübersetzt werden soll (siehe [Abschnitt „Vorübersetzung steuern“ auf Seite 93](#)).
- (6) Bibliothek, aus der der Kommunikationsbereich eingefügt wird (siehe [Abschnitt „INCLUDE-Bibliotheken angeben“ auf Seite 89](#)).
- (7) Einsprungsnamen für das vom ESQL-Precompiler erzeugte SQL-Bindelademodul festlegen (siehe [Abschnitt „Vorübersetzung steuern“ auf Seite 93](#)).
- (8) Voreingestellten Datenbanknamen und Schemanamen festlegen sowie Berechtigungsschlüssel angeben (siehe [Abschnitt „Eigenschaften des ESQL-COBOL-Programms festlegen“ auf Seite 98](#)).

## 3.2 ESQL-Precompiler-Optionen

Arbeitsweise und Ablauf des ESQL-Precompiler steuern Sie mit ESQL-Precompiler-Optionen. ESQL-Precompiler-Optionen werden im SDF-Format angegeben.

Dieser Abschnitt gibt einen inhaltlichen Überblick über die Steuerungsmöglichkeiten und beschreibt dann die einzelnen ESQL-Precompiler-Optionen in alphabetischer Reihenfolge.

### 3.2.1 Überblick über die ESQL-Precompiler-Optionen

In den folgenden Tabellen sind die ESQL-Precompiler-Optionen inhaltlich zu folgenden Gruppen zusammengefasst:

- Eingabequellen festlegen
- Eigenschaften des ESQL-COBOL-Programms festlegen
- Vorübersetzung steuern
- Ausgabeziele festlegen
- Jobvariablen angeben

#### 3.2.1.1 Eingabequellen festlegen

Precompiler-Option **INCLUDE-LIBRARY**

Parameter	Kurzbeschreibung	mögliche Angaben
	INCLUDE-Bibliothek	PLAM-Bibliotheken

Tabelle 7: Precompiler-Option INCLUDE-LIBRARY

Precompiler-Option **SOURCE**

Parameter	Kurzbeschreibung	mögliche Angaben
	ESQL-COBOL-Programm angeben	<ul style="list-style-type: none"> <li>– *SYSDTA</li> <li>– katalogisierte Datei</li> <li>– Bibliothekselement</li> </ul>

Tabelle 8: Precompiler-Option SOURCE

### 3.2.1.2 Eigenschaften des ESQL-COBOL-Programms festlegen

#### Precompiler-Option **SOURCE-PROPERTIES**

Parameter	Kurzbeschreibung	mögliche Angaben
HOST-LANGUAGE	Wirtssprache angeben	– COBOL
ESQL-DIALECT	zulässigen Sprachumfang festlegen	– SQL-Norm mit SESAM-spezifischen Erweiterungen – SQL-Norm – SESAM-SQL V1.1
QUOTATION-CHARACTER	Begrenzer für Zeichenketten festlegen	Anführungszeichen – einfache – doppelte
CATALOG	voreingestellten Datenbanknamen festlegen	– per Option – per eingebetteter Option im ESQL-COBOL-Programm
SCHEMA	voreingestellten Schemanamen festlegen	– per Option – per eingebetteter Option im ESQL-COBOL-Programm
AUTHORIZATION	Zugriffsberechtigung für die SQL-Anweisung festlegen	– per Option – per SQL-Anweisung
REPLACE-BY-FILE REPLACE-TOKENS	Ersetzungen für die Umstellung von vorhandenen Anwendungen auf die aktuelle SESAM/SQL-Version	Ersetzungen – aus Datei einlesen – als Liste angeben

Tabelle 9: Precompiler-Option SOURCE-PROPERTIES

### 3.2.1.3 Vorübersetzung steuern

#### Precompiler-Option **PRECOMPILER-ACTION**

Parameter	Kurzbeschreibung	mögliche Angaben
DATABASE-CONTACT	Datenbankkontakt bei Vorübersetzung	– mit Datenbankkontakt – ohne Datenbankkontakt
SQL-ENTRY-NAME	Einsprunghname des SQL-Bindelademoduls festlegen	entsprechender Einsprunghname
ESQL-STATEMENTS	Darstellung von SQL-Anweisungen im erzeugten COBOL-Programm	– als Kommentar – Entfernung aus dem erzeugten COBOL-Programm
ESQL-XREF	Querverweisliste als Kommentar erzeugen	– ja – nein

Tabelle 10: Precompiler-Option PRECOMPILER-ACTION

#### Ausgabeziele festlegen

#### Precompiler-Option **HOST-PROGRAM**

Parameter	Kurzbeschreibung	mögliche Angaben
	Ausgabeziel für COBOL-Programm festlegen	– katalogisierte Datei – Bibliothekselement

Tabelle 11: Precompiler-Option HOST-PROGRAM

#### Precompiler-Option **MODULE-LIBRARY**

Parameter	Kurzbeschreibung	mögliche Angaben
	Ausgabeziel für SQL-Bindelademodul festlegen	Bibliothekselement

Tabelle 12: Precompiler-Option MODULE-LIBRARY

#### Jobvariablen angeben

#### Precompiler-Option **MONJV**

Parameter	Kurzbeschreibung	mögliche Angaben
	Jobvariablen zur Überwachung des ESQL-Precompiler angeben	gewünschte Jobvariable

Tabelle 13: Precompiler-Option MONJV

### 3.2.2 Ausgabeziel für das erzeugte COBOL-Programm festlegen

Mit der Precompiler-Option **HOST-PROGRAM** geben Sie das Ausgabeziel für das vom ESQL-Precompiler erzeugte COBOL-Programm an.

HOST-PROGRAM
<pre> <b>HOST-PROGRAM</b> = <b>*STD-FILE</b> / &lt;full-filename 1..54 without-gen-vers&gt; / <b>*LINK(...)</b> /                 <b>*LIBRARY-ELEMENT(...)</b>  <b>*LINK(...)</b>         <b>LINK-NAME</b> = &lt;full-filename 1..8 without-gen-vers&gt; <b>*LIBRARY-ELEMENT(...)</b>         <b>LIBRARY</b> = &lt;full-filename 1..54 without-gen-vers&gt; / <b>*LINK(...)</b>                 <b>*LINK(...)</b>                     <b>LINK-NAME</b> = &lt;full-filename 1..8 without-gen-vers&gt;         ,<b>ELEMENT</b> = &lt;full-filename 1..38 without-cat-user-gen-vers&gt; (...)         <b>VERSION</b> = <b>*UPPER-LIMIT</b> / <b>*HIGHEST-EXISTING</b> / <b>*INCREMENT</b> / &lt;text 1..24&gt; </pre>

#### **HOST-PROGRAM =**

Ausgabeziel für das vom ESQL-Precompiler erzeugte COBOL-Programm angeben.

#### **HOST-PROGRAM = \*STD-FILE**

Das erzeugte COBOL-Programm wird in die katalogisierte Datei HOST.PROGRAM geschrieben.

#### **HOST-PROGRAM = <full-filename 1..54 without-gen-vers>**

Das COBOL-Programm wird in einer katalogisierten Datei mit dem angegebenen Namen gespeichert.

#### **HOST-PROGRAM = \*LINK(...)**

##### **LINK-NAME = <full-filename 1..8 without-gen-vers>**

Das erzeugte COBOL-Programm wird in die mit dem Linknamen zugewiesene katalogisierte Datei geschrieben.

#### **HOST-PROGRAM = \*LIBRARY-ELEMENT(...)**

Das erzeugte COBOL-Programm wird in dem angegebenen Element der PLAM-Bibliothek gespeichert.

##### **LIBRARY = <full-filename 1..54 without-gen-vers>**

Name der PLAM-Bibliothek, in der das erzeugte COBOL-Programm gespeichert wird.

##### **LIBRARY = \*LINK(...)**

Die PLAM-Bibliothek für das erzeugte COBOL-Programm wird über einen Linknamen zugewiesen.

**LINK-NAME = <full-filename 1..8 without-gen-vers>**

Linkname der PLAM-Bibliothek für das erzeugte COBOL-Programm.

**ELEMENT = <full-filename 1..38 without-cat-user-gen-vers>(…)**

Bibliothekselement, in dem das COBOL-Programm gespeichert wird. cat und user können angegeben werden, werden aber nicht als solche interpretiert.

**VERSION =**

Version des Bibliothekselements angeben.

**VERSION = \*UPPER-LIMIT**

Das Bibliothekselement erhält die höchstmögliche Version (vom LMS mit @ angezeigt).

**VERSION = \*HIGHEST-EXISTING**

Das Bibliothekselement mit der höchsten Version wird überschrieben.

**VERSION = \*INCREMENT**

Die höchste vorhandene Versionsnummer wird um eins hochgezählt und dem Bibliothekselement zugeordnet. Voraussetzung dafür ist, dass die höchste vorhandene Version mit einer Ziffer endet. Sonst erhalten Sie eine Fehlermeldung.

**VERSION = <alphanum-name 1..24>**

Das Bibliothekselement erhält die angegebene Versionsbezeichnung. Soll die Versionsbezeichnung hochgezählt werden können, muss mindestens das letzte Zeichen der Versionsbezeichnung eine Ziffer sein.

### 3.2.3 INCLUDE-Bibliotheken angeben

Mit der Precompiler-Option **INCLUDE-LIBRARY** geben Sie Bibliotheken an, in denen INCLUDE-Elemente gesucht werden.

<b>INCLUDE-LIBRARY</b>
<b>INCLUDE-LIBRARY = <u>*NONE</u></b> / list-poss(9): <b>*LINK(...)</b> / list-poss(9): <full-filename 1..54 without-gen-vers> <b>*LINK(...)</b>   <b>LINK-NAME</b> = <full-filename 1..8 without-gen-vers>

#### **INCLUDE-LIBRARY =**

PLAM-Bibliotheken angeben, aus denen INCLUDE-Elemente eingefügt werden können. Die Bibliotheken werden in der angegebenen Reihenfolge durchsucht. Sie können bis zu 9 Bibliotheken angeben.

#### **INCLUDE-LIBRARY = \*NONE**

Es können keine INCLUDE-Elemente eingefügt werden.

#### **INCLUDE-LIBRARY = list-poss (9): \*LINK(...)**

##### **LINK-NAME = <full-filename 1..8 without-gen-vers>**

Liste von Verweisen auf Bibliotheken, die nach INCLUDE-Elementen durchsucht werden.

#### **INCLUDE-LIBRARY = list-poss(9): <full-filename 1..54 without-gen-vers>**

Liste von Bibliotheken, die nach INCLUDE-Elementen durchsucht werden.

### 3.2.4 Ausgabeziel für SQL-Bindelademodul festlegen

Mit der Precompiler-Option **MODULE-LIBRARY** geben Sie die Bibliothek an, in die der ESQL-Precompiler das bei der Vorübersetzung erzeugte SQL-Bindelademodul schreibt.

<b>MODULE-LIBRARY</b>
<b>MODULE-LIBRARY = *<u>LIBRARY-ELEMENT</u>(...)</b> <b>*LIBRARY-ELEMENT(...)</b> <b>LIBRARY = &lt;full-filename 1..54 without-gen-vers&gt; / *LINK(...)</b> <b>*LINK(...)</b> <b>LINK-NAME = &lt;full-filename 1..8 without-gen-vers&gt;</b> <b>,ELEMENT = &lt;full-filename 1..38 without-cat-gen-user-vers&gt;(…)</b> <b>VERSION = *<u>UPPER-LIMIT</u> /*HIGHEST-EXISTING / *INCREMENT /&lt;text 1..24&gt;</b>

#### **MODULE-LIBRARY = \*LIBRARY-ELEMENT(...)**

PLAM-Bibliothek angeben.

##### **LIBRARY = <full-filename 1..54 without-gen>**

Name der PLAM-Bibliothek für das erzeugte SQL-Bindelademodul. Existiert die Bibliothek noch nicht, wird sie automatisch angelegt. Voreingestellt ist die Bibliothek SQLPROG.PLIB.

##### **LIBRARY = \*LINK(...)**

PLAM-Bibliothek für das erzeugte SQL-Bindelademodul über Linknamen zuweisen.

##### **LINK-NAME = <full-filename 1..8 without-gen>**

Linkname der PLAM-Bibliothek für das erzeugte SQL-Bindelademodul.

##### **ELEMENT = <full-filename 1..38 without-cat-user-gen-vers>(…)**

Name des Bibliothekselements, in das der ESQL-Precompiler das erzeugte SQL-Bindelademodul schreibt. Voreingestellt ist SQLPROG.OUT.  
cat und user können angegeben werden, werden aber nicht als solche interpretiert.

##### **VERSION =**

Version des Bibliothekselements angeben.

##### **VERSION = \*UPPER-LIMIT**

Das Bibliothekselement erhält die höchstmögliche Version (vom LMS mit @ angezeigt).

##### **VERSION = \*HIGHEST-EXISTING**

Das Bibliothekselement mit der höchsten Version wird überschrieben.

**VERSION = \*INCREMENT**

Die höchste vorhandene Versionsnummer wird um eins hochgezählt und dem Bibliothekselement zugeordnet. Voraussetzung dafür ist, dass die höchste vorhandene Version mit einer Ziffer endet. Sonst erhalten Sie eine Fehlermeldung.

**VERSION = <alphanum-name 1..24>**

Das Element erhält die angegebene Versionsbezeichnung. Soll die Versionsbezeichnung hochgezählt werden können, muss mindestens das letzte Zeichen der Versionsbezeichnung eine Ziffer sein.

### 3.2.5 Jobvariable angeben

Die Precompiler-Option **MONJV** ist nur erlaubt, wenn Sie den ESQL-Precompiler mit dem Kommando START-ESQLCOB starten (siehe [Abschnitt „ESQL-Precompiler starten“ auf Seite 81](#)).

Mit der Precompiler-Option MONJV geben Sie eine Jobvariable zur Überwachung der Vorübersetzung an. Während der Vorübersetzung setzt der ESQL-Precompiler in der Jobvariablen eine Zustandsanzeige und einen Rückgabewert. Für eine Beschreibung der möglichen Zustandsanzeigen und Rückgabewerte siehe [Abschnitt „Überwachen des Beendungsverhaltens mit Jobvariablen“ auf Seite 105](#).

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

#### **MONJV = \*NONE**

Die Vorübersetzung wird nicht mit einer Jobvariable überwacht.

#### **MONJV = <full-filename 1..54 without-gen-vers>**

Gibt die Jobvariable an, mit der die Vorübersetzung überwacht wird.

### 3.2.6 Vorübersetzung steuern

Mit der Precompiler-Option **PRECOMPILER-ACTION** steuern Sie die Vorübersetzung.

PRECOMPILER-ACTION
<pre> PRECOMPILER-ACTION = PARAMETERS(...) PARAMETERS(...)   DATABASE-CONTACT = <u>NO</u> / YES(...)     YES(...)       CATALOG-CHECKS = <u>STD</u> / DYNAMIC       APPLICATION-TYPE = <u>INDEPENDENT</u> / LINKEDIN   ,SQL-ENTRY-NAME = &lt;name 1..7&gt;   ,ESQL-STATEMENTS = <u>SOURCE-COMMENTS</u> / REMOVED   ,ESQL-XREF = <u>NO</u> / YES </pre>

#### PRECOMPILER-ACTION = PARAMETERS(...)

Verhalten des ESQL-Precompiler steuern.

##### **DATABASE-CONTACT =**

Angabe, ob das ESQL-COBOL-Programm mit oder ohne Datenbankkontakt übersetzt wird.

##### **DATABASE-CONTACT = NO**

Das ESQL-COBOL-Programm wird ohne Datenbankkontakt übersetzt. Das Datenbanksystem muss während der Vorübersetzung nicht aktiv sein.

##### **DATABASE-CONTACT = YES**

Das ESQL-COBOL-Programm wird mit Datenbankkontakt übersetzt. Das Datenbanksystem muss während der Vorübersetzung aktiv sein. Vor dem Aufruf des ESQL-Precompiler müssen Sie eine Konfigurationsdatei zuweisen (siehe [Abschnitt „ESQL-Precompiler starten“ auf Seite 81](#)).

##### **CATALOG-CHECKS =**

Wird das ESQL-COBOL-Programm mit Datenbankkontakt übersetzt, kann festgelegt werden, ob Widersprüche zwischen dem ESQL-COBOL-Programm und der Datenbank als Fehler oder als Warnung gewertet werden. Widersprüche können z.B. auftreten, wenn noch nicht alle vorgesehenen Tabellen einer Datenbank angelegt sind. Werden bei der Vorübersetzung nur Fehler der Fehlerklasse „Hinweis“ oder „Warnung“ erkannt, wird ein SQL-Bindelademodul erzeugt. Tritt ein Fehler der Fehlerklasse „Fehler“ auf, wird kein SQL-Bindelademodul erzeugt.

**CATALOG-CHECKS = STD**

Widersprüche zwischen dem ESQL-COBOL-Programm und der Datenbank werden als Fehler gewertet.

**CATALOG-CHECKS = DYNAMIC**

Widersprüche zwischen dem ESQL-COBOL-Programm und der Datenbank werden als Fehler der Fehlerklasse „Warnung“ gewertet. Das SQL-Bindelademodul wird auch erzeugt, wenn Widersprüche erkannt werden. Vor dem Starten der ESQL-COBOL-Anwendung müssen Sie die Datenbank oder das ESQL-COBOL-Programm entsprechend anpassen.

**APPLICATION-TYPE =**

Festlegen, ob bei der Vorübersetzung mit Datenbankkontakt der independent DBH oder der linked-in DBH verwendet wird. Informationen zum DBH finden Sie im „[Basishandbuch](#)“ [1] und im Handbuch „[Datenbankbetrieb](#)“ [5].

**APPLICATION-TYPE = INDEPENDENT**

Bei der Vorübersetzung mit Datenbankkontakt wird der independent DBH verwendet. Der independent DBH kann mit mehreren Auftraggebern zusammenarbeiten. Verwenden Sie den independent DBH, wenn die Anwendung mit anderen Anwendungen konkurrierend auf die Datenbank zugreift.

**APPLICATION-TYPE = LINKEDIN**

Bei der Vorübersetzung mit Datenbankkontakt wird der linked-in DBH verwendet. Der linked-in DBH kann nur mit einem Auftraggeber zusammenarbeiten. Verwenden Sie daher den linked-in DBH, wenn Sie auf Datenbanken zugreifen möchten, auf die keine anderen Anwendungen zugreifen, z.B. auf eine private Testdatenbank.

**SQL-ENTRY-NAME = <name 1..7>**

Einsprunghname, mit dem das SQL-Bindelademodul im COBOL-Programm aufgerufen wird. Den Einsprunghnamen müssen Sie angeben. Innerhalb einer Ausführungseinheit aus mehreren ESQL-COBOL-Programmen muss der Einsprunghname eindeutig sein. Aus Kompatibilitätsgründen zu ESQL-COBOL V1.1 hängt der ESQL-Precompiler an den Einsprunghnamen ein Q an.

**ESQL-STATEMENTS =**

Festlegen, ob SQL-Anweisungen im COBOL-Programm als Kommentare erhalten bleiben.

**ESQL-STATEMENTS = SOURCE-COMMENTS**

SQL-Anweisungen sind im COBOL-Programm als Kommentare enthalten.

**ESQL-STATEMENTS = REMOVED**

Die SQL-Anweisungen werden aus dem COBOL-Programm entfernt.

**ESQL-XREF =**

Legt fest, ob eine Querverweisliste in den Informationsteil am Ende des erzeugten COBOL-Programms aufgenommen werden soll. Die Querverweisliste enthält die verwendeten Datenbanknamen, Dateinamen und Benutzervariablen. Für jedes Vorkommen einer Benutzervariable wird die Zeilennummer des ESQL-COBOL-Programms, die Datei, die Verwendung der Benutzervariable (Eingabe, Ausgabe, Ein-/Ausgabe, definierend oder unbekannt) sowie die Zeilennummer im vom ESQL-Precompiler erzeugten COBOL-Programm ausgegeben.

### 3.2.7 Eingabequelle für den ESQL-Precompiler angeben

Mit der Precompiler-Option **SOURCE** geben Sie an, aus welcher Eingabequelle der ESQL-Precompiler das ESQL-COBOL-Programm liest.

SOURCE
<pre> <b>SOURCE</b> = <u>*SYSDTA</u> / &lt;full-filename 1..54 without-gen-vers&gt; / *<b>LINK</b>(...) / *<b>LIBRARY-ELEMENT</b>(...)  *<b>LINK</b>(...)     <b>LINK-NAME</b> = &lt;full-filename 1..8 without-gen-vers&gt;  *<b>LIBRARY-ELEMENT</b>(...)     <b>LIBRARY</b> = &lt;full-filename 1..54 without-gen-vers&gt; / *<b>LINK</b>(...)       *<b>LINK</b>(...)       <b>LINK-NAME</b> = &lt;full-filename 1..8 without-gen&gt;         <b>,ELEMENT</b> = &lt;full-filename 1..38 without-cat-user-gen-vers&gt; (...)         <b>VERSION</b> = <u>*HIGHEST-EXISTING</u> / <b>UPPER-LIMIT</b> / &lt;text 1..24&gt; </pre>

#### **SOURCE =**

Eingabequelle angeben.

#### **SOURCE = \*SYSDTA**

Der ESQL-Precompiler liest das ESQL-COBOL-Programm von der Systemdatei SYSDTA. Ist SYSDTA der Datensichtstation zugewiesen, können Sie das ESQL-COBOL-Programm im Dialog angeben. Der ESQL-Precompiler gibt in der ersten Spalte einen Stern \* als Eingabeaufforderung aus. Sie beenden die Eingabe des ESQL-COBOL-Programms wie folgt:

```

Taste 
/EOF
/RESUME-PROGRAM

```

Mit dem BS2000-Kommando ASSIGN-SYSDTA können Sie vor dem Aufruf des ESQL-Precompiler SYSDTA auf eine katalogisierte Datei oder eine PLAM-Bibliothek umlenken. Sie können dann das ESQL-COBOL-Programm sowie die Precompiler-Optionen aus der katalogisierten Datei oder dem Bibliothekselement einlesen. Es empfiehlt sich jedoch, die Precompiler-Optionen und das ESQL-COBOL-Programm getrennt zu speichern.

#### **SOURCE = <full-filename 1..54 without-gen-vers>**

Das ESQL-COBOL-Programm wird aus der angegebenen katalogisierten Datei gelesen.

#### **SOURCE = \*LINK(...)**

Das ESQL-COBOL-Programm wird aus einer katalogisierten Datei gelesen, die über Linknamen zugewiesen ist.

#### **LINK-NAME = <full-filename 1..8 without-gen-vers>**

Gibt den Linknamen an, über den die katalogisierte Datei zugewiesen ist.

**SOURCE = \*LIBRARY-ELEMENT(...)**

Angabe, aus welcher PLAM-Bibliothek und aus welchem Bibliothekselement das ESQL-COBOL-Programm gelesen wird.

**LIBRARY = <full-filename 1..54 without-gen-vers>**

Name der PLAM-Bibliothek.

**LIBRARY = \*LINK(...)**

Das ESQL-COBOL-Programm wird aus einer PLAM-Bibliothek eingelesen, die über Linknamen zugewiesen ist.

**LINK-NAME = <full-filename 1..8 without-gen-vers>**

Linkname der PLAM-Bibliothek.

**ELEMENT = <full-filename 1..38 without-cat-user-gen-vers>**

Name des Bibliothekselements. cat und user können angegeben werden, werden aber nicht als solche interpretiert.

**VERSION =**

Festlegen der Version des Bibliothekselements.

**VERSION = \*HIGHEST-EXISTING**

Der ESQL-Precompiler liest das ESQL-COBOL-Programm aus dem Bibliothekselement mit der höchsten Version.

**VERSION = \*UPPER-LIMIT**

Der ESQL-Precompiler liest das ESQL-COBOL-Programm aus dem Bibliothekselement mit der höchstmöglichen Version (vom LMS mit @ angezeigt).

**VERSION = <alphanum-name 1..24>**

Der ESQL-Precompiler liest das ESQL-COBOL-Programm aus dem Bibliothekselement mit der angegebenen Version.

### 3.2.8 Eigenschaften des ESQL-COBOL-Programms festlegen

Mit der Precompiler-Option **SOURCE-PROPERTIES** legen Sie Eigenschaften des ESQL-COBOL-Programms fest.

SOURCE-PROPERTIES
<pre> SOURCE-PROPERTIES = <u>PARAMETERS</u>(...) / STD PARAMETERS(...)   HOST-LANGUAGE = <u>COBOL</u>   ,ESQL-DIALECT = <u>ALL-FEATURES</u>(...) / ISO(...) / OLD(...)   (...)       UTM-RESTRICTIONS = <u>NO</u> / YES   ,QUOTATION-CHARACTER = <u>ESQL-STANDARD</u> / HOST-STANDARD   ,CATALOG = <u>*INLINE</u> / &lt;text 1..18&gt; / &lt;c-string 1..18&gt;   ,SCHEMA = <u>*INLINE</u> / &lt;text 1..31&gt; / &lt;c-string 1..31&gt;   ,AUTHORIZATION = <u>*IMPLICIT</u> / &lt;text 1..18&gt; / &lt;c-string 1..18&gt;   ,REPLACE-BY-FILE = <u>*NONE</u> / &lt;full-filename 1..54 without-gen-vers&gt; / *LINK(...) /     LIBRARY-ELEMENT(...)     *LINK(...)         LINK-NAME = &lt;full-filename 1..8 without-gen-vers&gt;     LIBRARY-ELEMENT(...)         LIBRARY = &lt;full-filename 1..54 without-gen&gt; / *LINK(...)       *LINK(...)           LINK-NAME = &lt;full-filename 1..8 without-gen-vers&gt;   ,ELEMENT = &lt;full-filename 1..38 without-cat-user-gen-vers(...)       VERSION = <u>HIGHEST-EXISTING</u> / UPPER-LIMIT / &lt;text 1..24&gt;   ,REPLACE-TOKENS = <u>*NONE</u> /list-poss(2000):*SUBSTITUTE(...)   SUBSTITUE(...)       IDENTIFIER = &lt;c-string 1..128&gt;     ,REPLACEMENT = &lt;c-string 1..128&gt; </pre>

#### SOURCE-PROPERTIES = PARAMETERS

Eigenschaften des ESQL-COBOL-Programms festlegen.

##### HOST-LANGUAGE = COBOL

Wirtssprache des ESQL-Programms.

**ESQL-DIALECT =**

Erlaubten Sprachumfang festlegen. Dieser Parameter erleichtert die Portierung von ESQL-COBOL-Programmen, die für frühere SESAM/SQL-Versionen oder andere SQL-Datenbanksysteme entwickelt wurden. Enthält das ESQL-Programm SQL-Elemente, die im angegebenen Sprachumfang nicht erlaubt sind, gibt der ESQL-Precompiler Fehlermeldungen aus und erzeugt kein SQL-Bindelademodul.

**ESQL-DIALECT = ALL-FEATURES(...)**

Sprachumfang der SQL-Norm ISO/IEC 9075:2003 und SESAM-spezifische Erweiterungen als zulässigen Sprachumfang festlegen. Diese Angabe empfiehlt sich, wenn alle Merkmale von SESAM/SQL verwendbar sein sollen.

**ESQL-DIALECT = ISO(...)**

SQL-Norm ISO/IEC 9075:2003 als zulässigen Sprachumfang festlegen. Diese Angabe empfiehlt sich bei der Vorübersetzung von ESQL-COBOL-Programmen, die für andere SQL-Datenbanksysteme entwickelt wurden. Mit dieser Angabe sind nur die in der SQL-Norm festgelegten Schlüsselwörter reserviert. Die Schlüsselwörter, die von SESAM/SQL zusätzlich reserviert werden, gelten mit dieser Angabe nicht als reserviert.

**ESQL-DIALECT = OLD(...)**

Sprachumfang von SESAM/SQL V1.1 als zulässigen Sprachumfang festlegen. Diese Angabe kann bei der Vorübersetzung von ESQL-COBOL-Programmen angegeben werden, die für SESAM/SQL V1.1 entwickelt wurden. Dadurch können reservierte Schlüsselwörter, die erst für die aktuelle SESAM/SQL-Version reserviert wurden, noch als Namen verwendet werden.

**UTM-RESTRICTIONS =**

Festlegen, ob ESQL-COBOL-Programm auf UTM-gerechten Sprachumfang geprüft wird. Im UTM-Betrieb sind zur Transaktionssicherung nur UTM-Sprachmittel erlaubt. Die SQL-Anweisungen COMMIT WORK und ROLLBACK WORK zum Beenden und Zurücksetzen von Transaktionen sind daher nicht erlaubt.

**UTM-RESTRICTIONS = NO**

Die SQL-Anweisungen COMMIT WORK und ROLLBACK WORK sind zugelassen.

**UTM-RESTRICTIONS = YES**

Die SQL-Anweisungen COMMIT WORK und ROLLBACK WORK sind nicht erlaubt. Bei Programmausführung ergeben die SQL-Anweisungen PREPARE und EXECUTE IMMEDIATE für COMMIT WORK und ROLLBACK WORK einen Fehler.

**QUOTATION-CHARACTER =**

Festlegen, wie alphanumerische Literale und Namen in Anführungszeichen in SQL-Anweisungen begrenzt werden. Für Literale im COBOL-Text werden die Begrenzer bei der Übersetzung mit dem COBOL-Compiler festgelegt (siehe COBOL2000 Handbuch „[Benutzerhandbuch](#)“ [17]).

**QUOTATION-CHARACTER = ESQL-STANDARD**

Anführungszeichen werden gemäß der SQL-Norm verwendet: Alphanumerische Literale werden mit einfachen Anführungszeichen begrenzt, Namen in Anführungszeichen mit doppelten Anführungszeichen.

**QUOTATION-CHARACTER = HOST-STANDARD**

Alphanumerische Literale werden mit doppelten Anführungszeichen begrenzt, Namen in Anführungszeichen mit einfachen Anführungszeichen.

**CATALOG =**

Voreinstellung für den Datenbanknamen festlegen. Dieser Datenbankname gilt nach den Regeln von SQL für alle SQL-Objekte des ESQL-COBOL-Programms, für die Sie keinen Datenbanknamen angegeben haben. Den voreingestellten Datenbanknamen müssen Sie explizit angeben, auch wenn Sie die Vorübersetzung ohne Datenbankkontakt durchführen.

**CATALOG = \*INLINE**

Die Voreinstellung für den Datenbanknamen wird mit der Precompiler-Option SOURCE-PROPERTIES direkt im ESQL-COBOL-Programm angegeben.

**CATALOG = <text 1..18>/<c-string 1..18>**

Angegebenen Datenbanknamen als voreingestellten Datenbanknamen festlegen. Der angegebene Datenbankname muss den Konventionen für Datenbanknamen entsprechen (siehe COBOL2000 Handbuch „[SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen](#)“ [2]).

**SCHEMA =**

Voreinstellung für den Schemanamen festlegen. Dieser Schemaname gilt nach den Regeln von SQL für alle SQL-Objekte des ESQL-COBOL-Programms, für die Sie keinen Schemanamen angegeben haben. Den voreingestellten Schemanamen müssen Sie explizit angeben, auch wenn Sie die Vorübersetzung ohne Datenbankkontakt durchführen.

**SCHEMA = \*INLINE**

Die Voreinstellung für den Schemanamen wird mit der Precompiler-Option SOURCE-PROPERTIES direkt im ESQL-COBOL-Programm angegeben.

**SCHEMA = <text 1..31> / <c-string 1..31>**

Angegebenen Schemanamen als voreingestellten Schemanamen festlegen. Der angegebene Schemanamen muss innerhalb der Datenbank eindeutig sein.

**AUTHORIZATION =**

Berechtigungsschlüssel festlegen.

**AUTHORIZATION = \*IMPLICIT**

Der Berechtigungsschlüssel wird mit der SQL-Anweisung SET SESSION AUTHORIZATION angegeben.

**AUTHORIZATION = <text 1..18>/<c-string 1..18>**

Festlegen des Berechtigungsschlüssels. Der hier festgelegte Berechtigungsschlüssel kann nicht mit der SQL-Anweisung SET SESSION AUTHORIZATION geändert werden.

**REPLACE-BY-FILE =**

Ersetzungsdatei angeben, aus der die Ersetzungspaare eingelesen werden. Die Ersetzungsdatei hat folgendes Format:

```
//PRECOMPILE SOURCE-PROPERTIES= ( -
// ( -
// REPLACE-TOKENS= -
// *SUBSTITUTE( 'zu_ersetzende_zeichenfolge1' , 'ersetzung1' ) -
// ,*SUBSTITUTE( 'zu_ersetzende_zeichenfolge2' , 'ersetzung2' ) -
//      ...
// ) )
// END
//PRECOMPILE SOURCE-PROPERTIES= -
// ( -
//      ...
// )
// END
```

In der der Datei dürfen mehrere PRECOMPILE-Anweisungen angegeben werden. Die END-Anweisung muss als letzte Anweisung in der Datei stehen. Um die Fehlerdiagnose zu erleichtern, empfiehlt es sich, Ersetzungspaare auf mehrere PRECOMPILE-Anweisungen zu verteilen.

*Beispiel*

Namen von Temporary Views und Tabellen mit dem Schemanamen qualifizieren. Im Beispiel wird davon ausgegangen, dass Anführungszeichen gemäß der SQL-Norm verwendet werden: Namen in Anführungszeichen werden mit doppelten Anführungszeichen begrenzt (Option SOURCE-PROPERTIES, Parameter QUOTATION-CHARACTER = ESQL-STANDARD).

```
//PRECOMPILE SOURCE-PROPERTIES= ( -
// ( -
// REPLACE-TOKENS= -
// *SUBSTITUTE( 'PERSONALDATEN' , 'MODULE.PERSONALDATEN' ) -
// ,*SUBSTITUTE( '"GEWINN&VERLUST"' , 'MODULE."GEWINN&VERLUST"' ) -
// ,*SUBSTITUTE( 'ARTIKEL' , 'TEILE_SCHEMA.ARTIKEL' ) -
// ))
// END
```

**Erläuterung**

Die Namen der Temporary Views PERSONALDATEN und GEWINN&VERLUST werden mit MODULE qualifiziert. GEWINN&VERLUST ist ein Name in Anführungszeichen. Die Tabelle ARTIKEL wird mit dem Schemanamen TEILE\_SCHEMA qualifiziert.

**REPLACE-BY-FILE = \*NONE**

Es wird keine Ersetzungsdatei angegeben.

**REPLACE-BY-FILE = <full-filename 1..54 without-gen-vers>**

Der ESQL-Precompiler verwendet als Ersetzungsdatei die angegebene katalogisierte Datei.

**REPLACE-BY-FILE = \*LINK(...)**

Der ESQL-Precompiler verwendet als Ersetzungsdatei eine katalogisierte Datei, die über Linknamen zugewiesen ist.

**LINK-NAME = <full-filename 1..8 without-gen-vers>**

Linkname der katalogisierten Datei.

**REPLACE-BY-FILE = \*LIBRARY-ELEMENT(...)**

Angabe, aus welcher PLAM-Bibliothek und aus welchem Bibliothekselement die Ersetzungspaare gelesen werden.

**LIBRARY = <full-filename 1..54 without-gen-vers>**

Name der PLAM-Bibliothek.

**LIBRARY = \*LINK(...)**

Der ESQL-Precompiler liest die Ersetzungspaare aus einer PLAM-Bibliothek, die über einen Linknamen zugewiesen ist.

**LINK-NAME = <full-filename 1..8 without-gen-vers>**

Linkname der PLAM-Bibliothek.

**ELEMENT = <full-filename 1..38 without-cat-user-gen-vers>(…)**

Name des Bibliothekselements. cat und user können angegeben werden, werden aber nicht als solche interpretiert.

**VERSION =**

Version des Bibliothekselements festlegen.

**VERSION = \*HIGHEST-EXISTING**

Der ESQL-Precompiler liest die Ersetzungspaare aus dem Bibliothekselement mit der höchsten Version.

**VERSION = \*UPPER-LIMIT**

Der ESQL-Precompiler liest die Ersetzungspaare aus dem Bibliothekselement mit der höchstmöglichen Version (vom LMS mit @ angezeigt).

**VERSION = <alphanum-name 1..24>**

Der ESQL-Precompiler liest die Synonyme aus dem angegebenen Bibliothekselement.

**REPLACE-TOKENS =**

Zur Umstellung und Portierung von ESQL-COBOL-Anwendungen Ersetzungen durchführen. Es können Namen, Namen in Anführungszeichen und Schlüsselwörter ersetzt werden. Die zu ersetzende Zeichenfolgen und die zugehörigen Ersetzungen können paarweise angegeben oder aus einer Ersetzungsdatei eingelesen werden. Alle Angaben (Angabe beim Aufruf des Precompilers, im ESQL-COBOL-Programm, in einer Ersetzungsdatei) werden gleichrangig ausgewertet.

Für die Ersetzung gelten folgende Regeln:

- Ersetzungen werden nur im SQL-Text durchgeführt. Der SQL-Text kann direkt im Text des ESQL-COBOL-Programms stehen oder mit der SQL-Anweisung INCLUDE eingefügt worden sein.
- Es können Namen, Namen in Anführungszeichen und Schlüsselwörter ersetzt werden.
- Die zu ersetzende Zeichenfolge muss vollständig in einer Zeile stehen.
- Welche Begrenzer bei Namen in Anführungszeichen gelten (einfache oder doppelte Anführungszeichen), legt der Parameter QUOTATION-CHARACTER fest. Anführungszeichen, die nicht als Begrenzer, sondern als alphanumerische Literale interpretiert werden sollen, müssen verdoppelt werden.

- Es erfolgt keine Ersetzung in
  - den SQL-Anweisungen INCLUDE oder WHENEVER
  - Zeichenfolgen, die der Angabe EXEC SQL folgen und mit INCLUDE oder WHENEVER beginnen
  - den Zeichenketten EXEC SQL bzw. END-EXEC
  - numerischen Literalen
  - alphanumerischen Literalen
  - Namen von Benutzervariablen

**REPLACE-TOKENS = \*NONE**

Es werden keine Ersetzungen durchgeführt.

**REPLACE-TOKENS = list-poss(2000):\*SUBSTITUTE(...)**

Ersetzungspaare angeben. Anführungszeichen, die nicht als Begrenzer, sondern als alphanumerische Literale interpretiert werden sollen, müssen verdoppelt werden. Doppelt angegebene Anführungszeichen gelten als zwei Zeichen.

**IDENTIFIER = <c-string 1..128>**

Zeichenfolgen, die ersetzt werden sollen. Sie können Namen, Namen in Anführungszeichen und Schlüsselwörter angeben.

**REPLACEMENT = <c-string 1..128>**

Ersetzungen für die mit IDENTIFIER angegebenen Zeichenfolgen.

**SOURCE-PROPERTIES = STD**

Für alle Parameter gelten die Voreinstellungen.

### 3.3 Beendungsverhalten des ESQL-Precompiler

Das Beendungsverhalten des Precompiler hängt davon ab, ob während der Vorübersetzung Fehler auftraten und zu welcher Fehlerklasse die aufgetretenen Fehler gehören. Dieser Abschnitt beschreibt die Überwachung des Beendungsverhaltens mit Jobvariablen, den Aufbau und die Ausgabe von Fehlermeldungen sowie das Erstellen von Diagnoseunterlagen.

#### 3.3.1 Überwachen des Beendungsverhaltens mit Jobvariablen

Das Beendungsverhalten ist vor allem von Bedeutung, wenn Sie den ESQL-Precompiler in einer Prozedur aufrufen. Sie können den Ablauf und das Beendungsverhalten des Precompiler mit Jobvariablen überwachen.

Die folgende Tabelle gibt einen Überblick über die möglichen Beendungszustände, deren Auswirkungen auf den weiteren Ablauf einer Prozedur und den Inhalt einer Jobvariablen.

Fehler	Beendigung	Jobvariable		Verhalten in Prozeduren
		Zustandsanzeige	Rückgabewert	
keine Fehler	normal SQL-Bindelademo- dual wird erzeugt	\$T	0000	keine Verzwei- gung
Meldungen der Fehler- klasse "Hinweis"			0001	
Meldungen der Fehler- klasse "Warnung"			1002	
ESQL-Precompiler ist fehlerfrei gelaufen, aber Meldungen der Fehlerklasse "Fehler"	fehlerhaft Es wird keine SQL- Bindelademo- dual erzeugt		2004	Verzweigung zum nächsten STEP-, ABEND-, AB- ORT-, END- oder LOGOFF- Kommando
Fehler, z.B.: fehlerhaf- te Option, Eingabeda- tei existiert nicht, Spei- cherplatzmangel			2005	
Interner Fehler des ESQL-Precompiler			\$A	

Tabelle 14: Beendungsverhalten des ESQL-Precompiler

### 3.3.2 Meldungen des ESQL-Precompiler

Das ESQL-COBOL-System gibt Meldungen in der Regel auf SYSOUT und im vorübersetzten COBOL-Programm aus. Informationsmeldungen wie zum Beispiel Meldungen am Anfang oder Ende der Vorübersetzung werden nur auf SYSOUT ausgegeben. Wenn aufgrund des Fehlerzustands keine Ausgabe im COBOL-Programm möglich ist, werden Meldungen zu schwerwiegenden internen Fehlern oder Systemfehlern nur auf SYSOUT ausgegeben.

Die Meldungszeile steht im COBOL-Programm unterhalb der Zeile, in der der Fehler aufgetreten ist.

Meldungen des ESQL-COBOL-Systems haben folgenden Aufbau:

*Meldungsnummer Position Fehlerklasse - Meldungstext*

*Meldungsnummer*

Eindeutige Meldungsnummer, anhand der Sie den Meldungstext sowie Bedeutungs- und Maßnahme-Texte in [Kapitel „Meldungen des ESQL-COBOL-Systems“](#) nachschlagen können. Die Meldungen sind in folgende Nummernkreise unterteilt:

<b>Nummernkreis</b>	<b>Bedeutung</b>
IQB0000 - IQB0899	Meldungen zu Precompiler-Optionen, zur Ein- und Ausgabe, zur Speicherverwaltung und zur Modulerzeugung
IQB0900 - IQB0999	Interne Fehler des ESQL-Precompiler
IQB1000 - IQB1999	Meldungen der syntaktischen Analyse
IQB2000 - IQB2999	Meldungen der semantischen Analyse
IQB3000 - IQB3099	ESQL-konstruktsspezifische Meldungen
IQB9000 - IQB9099	Informationsmeldungen

*Position* Fehlerposition im Format *zzzzz:ddd*.

*zzzzz* Nummer der Zeile, in der der Fehler aufgetreten ist.

*ddd* Datei, die den Fehler enthält. 1 steht für das ESQL-COBOL-Programm. INCLUDE-Dateien werden durchnummeriert. Die INCLUDE-Dateien mit den zugehörigen Nummern werden im Informationsteil des erzeugten COBOL-Programms aufgelistet.

Treten Fehler auf, die keiner einzelnen Zeile zugeordnet werden können, entfällt die Positionsangabe.

<i>Fehlerklasse</i>	Angabe, wie schwerwiegend der Fehler ist. Die Fehlerklassen haben folgende Bedeutung:
N	<b>Hinweis (Note)</b> Es liegt kein schwerwiegender Fehler vor. Es empfiehlt sich jedoch, die entsprechende Stelle des ESQL-COBOL-Programms zu überprüfen. Die Vorübersetzung wird fortgesetzt, das SQL-Bindelademodul wird erzeugt.
W	<b>Warnung</b> Der Precompiler hat eine Inkonsistenz oder eine Angabe, die in einer zukünftigen Version nicht mehr unterstützt wird, gefunden. Der aufgetretene Fehler hat keine unmittelbaren oder schwerwiegenden Folgen. Es empfiehlt sich jedoch, die entsprechende Stelle des ESQL-COBOL-Programms zu überprüfen. Die Vorübersetzung wird fortgesetzt, das SQL-Bindelademodul kann erzeugt werden.
E	<b>Fehler (Error)</b> Der ESQL-Precompiler hat einen z.B. lexikalischen, syntaktischen oder semantischen Fehler gefunden. Die Vorübersetzung wird fortgesetzt, es wird jedoch kein SQL-Bindelademodul erzeugt. Korrigieren Sie das ESQL-COBOL-Programm und starten Sie erneut die Vorübersetzung.
F	<b>Abbruch (Fatal)</b> Ein schwerwiegender Fehler führte zum Abbruch der Vorübersetzung. Dabei kann es sich um Fehler in den Precompiler-Optionen oder Systemfehler im ESQL-COBOL-System handeln. Es wird kein SQL-Bindelademodul erzeugt. Korrigieren Sie den entsprechenden Fehler und starten Sie die Vorübersetzung erneut.
S	<b>System</b> Interner Fehler des ESQL-Precompiler. Es wird kein SQL-Bindelademodul erzeugt. Erstellen Sie die erforderlichen Diagnoseunterlagen und verständigen Sie den zuständigen Systemdienst (siehe <a href="#">Abschnitt „Diagnoseunterlagen erstellen“ auf Seite 108</a> ).
<i>Meldungstext</i>	Der Meldungstext beschreibt den aufgetretenen Fehler.

Am Ende des COBOL-Programms gibt der ESQL-Precompiler zusätzlich folgende Informationen aus:

- Gesamtzahl der aufgetretenen Fehler
- Anzahl der Fehler der einzelnen Fehlerklassen
- Gesetzte Precompiler-Optionen
- Zeilennummern der Verweise auf Datenbanknamen im ESQL-COBOL-Programm und im COBOL-Programm

- Namen und Nummer des ESQL-COBOL-Programms und der verwendeten INCLUDE-Dateien sowie die Zeilennummern des ESQL-COBOL-Programms und des COBOL-Programms, in der die INCLUDE-Dateien referenziert bzw. eingebunden werden.
- Querverweisliste mit den Namen aller Benutzervariablen und der Art der Referenzierung, falls bei der Precompiler-Option PRECOMPILER-ACTION der Parameter XREF=YES gesetzt wurde (siehe [Abschnitt „Vorübersetzung steuern“ auf Seite 93](#)).

Eine Liste aller Meldungen finden Sie im [Kapitel „Meldungen des ESQL-COBOL-Systems“ auf Seite 141](#).

### 3.3.3 Diagnoseunterlagen erstellen

Um im Fehlerfall die erforderlichen Diagnoseunterlagen zu erhalten, empfiehlt es sich, den Auftragsablauf von ESQL-COBOL-Übersetzungsläufen zu protokollieren. Dazu verwenden Sie z.B. das BS2000-Kommando:

```
MODIFY-JOB-OPTIONS LOGGING=PARAMETERS(LISTING=YES)
```

Bei internen Fehlern des ESQL-Precompiler (Nummernkreis IQB0900 - IQB0999) erstellen Sie folgende Diagnoseinformationen für den zuständigen Systemdienst:

- verwendetes ESQL-COBOL-Programm
- erzeugtes COBOL-Programm, falls vorhanden
- Versionsnummer des ESQL-Precompiler
- Versionsnummer der CRTE und des SQL-Laufzeitsystems bzw. des SESAM/SQL-Server
- Dump, falls angeboten
- Liste der gesetzten ESQL-Precompiler-Optionen
- Protokoll des Auftragsablaufs

Die Versionsnummer des ESQL-Precompiler wird beim Starten des ESQL-Precompiler am Bildschirm ausgegeben. Außerdem wird sie in das erzeugte COBOL-Programm geschrieben. Die Versionsnummer der CRTE und des SQL-Laufzeitsystems können Sie bei Ihrem Systemverwalter erfragen. Die verwendeten ESQL-Precompiler-Optionen sind am Ende des erzeugten COBOL-Programms aufgelistet. Haben Sie mit dem BS2000-Kommando MODIFY-JOB-OPTIONS die Protokollierung des Auftragsablaufs festgelegt, wird das Protokoll auf SYSLST geschrieben.

---

## 4 COBOL-Programm übersetzen

Zum Übersetzen des vom ESQL-Precompiler erzeugten COBOL-Programms verwenden Sie den COBOL-Compiler COBOL2000.



Wenn Sie neue Sprachelemente, die erst mit Version V3.0 implementiert wurden, zusammen mit dem COBOL85 Compiler verwenden, können Inkompatibilitäten auftreten.

Die Beschreibung des COBOL-Compiler finden Sie in den COBOL2000 Handbüchern „[Sprachbeschreibung](#)“ [16] und „[Benutzerhandbuch](#)“ [17].

### Kommandofolge zum Übersetzen eines COBOL-Programms

```
/START-COBOL2000-COMPILER - (1)
/SOURCE=cobol-programm - (2)
/ ,COMPILER-ACTION=MODULE-GENERATION (3)
```

- (1) COBOL-Compiler laden und starten. Die Eingaben werden von der Datensichtstation gelesen.
- (2) Zu übersetzendes COBOL-Programm zuweisen. Geben Sie das mit der Precompiler-Option HOST-PROGRAM festgelegte Ausgabeziel des ESQL-Precompiler an. Wenn Sie kein Ausgabeziel festgelegt haben, geben Sie das voreingestellte Ausgabeziel an: HOST.PROGRAM
- (3) Übersetzungsschritte festlegen. Der COBOL-Compiler führt eine vollständige Übersetzung durch. Nach fehlerfreier Übersetzung wird ein COBOL-Bindemodul erzeugt.



---

## 5 ESQL-COBOL-Anwendung binden

Zum Binden benötigen Sie den Binder BINDER und den dynamischen Bindelader DBL. Eine ausführliche Beschreibung des BINDER und des DBL finden Sie in den Handbüchern „BINDER“ [19] und „Bindelader-Starter in BS2000/OSD“ [20].

Beim Binden werden folgende Module zu einer ablauffähigen Einheit in Form einer Ladeeinheit gebunden:

- vom ESQL-Precompiler erzeugtes SQL-Bindelademodul
- vom COBOL-Compiler erzeugtes COBOL-Bindemodul
- SESAM/SQL-Konnektionsmodul für den independent oder linked-in DBH

Das COBOL-Laufzeitsystem kann mit der ESQL-COBOL-Anwendung zusammengebunden werden. Es empfiehlt sich jedoch, das Laufzeitsystem beim Starten der ESQL-COBOL-Anwendung dynamisch nachzuladen. Dadurch wird die ESQL-COBOL-Anwendung kleiner, und es wird beim Starten der ESQL-COBOL-Anwendung jeweils das aktuelle Laufzeitsystem verwendet.

### ESQL-COBOL-Anwendung binden

```
//START-BINDER (1)
//START-LLM-CREATION INTERNAL-NAME=interner_name (2)
//INCLUDE-MODULES LIBRARY=bibliothek, ELEMENT=(sql-bindelademodul) (3)
//INCLUDE-MODULES LIBRARY=bibliothek, ELEMENT=(cobol-bindemodul) (4)
//INCLUDE-MODULES LIBRARY=SYS.MOD, ELEMENT=sesam/sql-konnektionsmodul (5)
//RESOLVE-BY-AUTOLINK LIBRARY=crte-lib (6)
//SAVE-LLM LIBRARY=bibliothek, ELEMENT=element (7)
//END
```

- (1) BINDER aufrufen.
- (2) Bindelademodul (LLM) mit internem Namen erzeugen.

- (3) SQL-Bindelademodul einbinden. Geben Sie die Namen der Bibliothek und des Elements an, die Sie bei der Vorübersetzung mit der Precompiler-Option MODULE-LIBRARY als Ausgabeziel für das SQL-Bindelademodul festgelegt haben. Wenn Sie kein Ausgabeziel festgelegt haben, geben Sie das voreingestellte Ausgabeziel an:  
LIBRARY=SQLPROG.PLIB, ELEMENT=SQLPROG.OUT
- (4) COBOL-Bindemodul einbinden. Geben Sie die Namen der Bibliothek und des Elements an, die Sie bei der Übersetzung mit dem COBOL-Compiler als Ausgabeziel festgelegt haben. Wenn Sie kein Ausgabeziel festgelegt haben, geben Sie das voreingestellte Ausgabeziel an:  
LIBRARY=\*OMF, ELEMENT=\*ALL
- (5) SESAM/SQL-Konnektionsmodul aus der SESAM/SQL-Modulbibliothek einbinden. Soll die ESQL-COBOL-Anwendung mit dem independent DBH zusammenarbeiten, geben Sie als Element SESMOD an. Für den linked-in DBH geben Sie das Element SESLINK an.
- (6) CRTE für das COBOL-Laufzeitsystem einbinden. Diese Anweisung muss nur gegeben werden, wenn das Laufzeitsystem statisch gebunden werden soll. Es empfiehlt sich jedoch, das Laufzeitsystem beim Starten der der ESQL-COBOL-Anwendung dynamisch nachzuladen (siehe [Kapitel „ESQL-COBOL-Anwendung starten“ auf Seite 113](#)).
- (7) Erzeugtes Bindelademodul unter dem angegebenen Namen als Element vom Typ L in der angegebenen Bibliothek speichern.

### Ausführungseinheit aus mehreren ESQL-COBOL-Anwendungen

Sie können auch eine Ausführungseinheit (RUN UNIT) aus mehreren übersetzten und ablauffähigen ESQL-COBOL-Anwendungen binden. Dabei ist folgendes zu beachten:

- Cursor und Temporary Views sind nur innerhalb des Moduls gültig, in dem sie definiert wurden, nicht in der gesamten Ausführungseinheit.
- Die Einsprungnamen, die Sie mit der Precompiler-Option PRECOMPILER-ACTION festgelegt haben, müssen innerhalb der Ausführungseinheit eindeutig sein (siehe [Abschnitt „Vorübersetzung steuern“ auf Seite 93](#)).

---

## 6 ESQ-LOBOL-Anwendung starten

Bevor Sie eine ESQ-LOBOL-Anwendung starten, müssen Sie die SESAM/SQL-Modulbibliothek dem Linknamen SESAMOML zuweisen. Es empfiehlt sich, das COBOL-Laufzeitsystem beim Starten dynamisch nachzuladen. Dazu weisen Sie die CRTE-Bibliothek als BLSLIB zu. Arbeitet die ESQ-LOBOL-Anwendung mit dem linked-in DBH zusammen, muss die CRTE-Bibliothek als BLSLIB zugewiesen sein und beim Kommando START-PROGRAM muss PROGRAM-MOD=ANY angegeben werden.

Sie können eine Konfigurationsdatei erstellen und zuweisen. Wenn Sie keine Konfigurationsdatei zuweisen, gelten für den Namen des DBH und der Konfiguration die voreingestellten Werte. Eine ausführliche Beschreibung der Konfigurationsdatei für den independent und den linked-in DBH finden Sie im „[Basishandbuch](#)“ [1].

**Kommandofolge zum Starten einer ESQL-COBOL-Anwendung**

```

/SET-FILE-LINK LINK-NAME=SESAMOML, FILE-NAME=sesam-modlib (1)
/SET-FILE-LINK LINK-NAME=BLSLIBnn, FILE-NAME=crte-lib (2)
/CONNECT-SESAM-CONFIGURATION TO-FILE=globale konfigurationsdatei, -
/ CONFIGURATION-LINK=linkname
oder
/SET-FILE-LINK LINK-NAME=SESCONF, FILE-NAME=dateiname (3)
/START-PROGRAM FROM-FILE=- (4)
/ *MODULE(LIBRARY=bibliothek, ELEMENT=element - (5)
/ ,PROGRAM-MODE=ANY - (6)
/ ,RUN-MODE=ADVANCED - (7)
/ (ALTERNATE-LIBRARIES =YES) - (8)
/ )

```

- (1) SESAM/SQL-Modulbibliothek zuweisen.
- (2) CRTE-Bibliothek als BLSLIB zuweisen. Es empfiehlt sich, BLSLIB00 zu verwenden.  
Diese Zuweisung ist nur erforderlich, wenn die Anwendung mit dem linked-in DBH zusammenarbeiten soll oder das COBOL-Laufzeitsystem dynamisch nachgeladen werden soll. Damit Module aus der CRTE-Bibliothek nachgeladen werden können, muss beim START-PROGRAM-Kommando ALTERNATE-LIBRARIES=YES angegeben werden.
- (3) Konfigurationsdatei zuweisen.
- (4) Lädt und startet die ESQL-COBOL-Anwendung.
- (5) Ruft den dynamischen Bindelader DBL zum Starten der ESQL-COBOL-Anwendung auf. Die ESQL-COBOL-Anwendung wird aus der angegebenen Bibliothek bzw. dem Element geladen. Geben Sie den Namen der Bibliothek und des Elements an, den Sie beim Binden der ESQL-COBOL-Anwendung bei SAVE-LLM als Ausgabeziel für die ESQL-COBOL-Anwendung angegeben haben.
- (6) Diese Angabe ist erforderlich, wenn die ESQL-COBOL-Anwendung mit dem linked-in DBH zusammenarbeitet.
- (7) Stellt den Betriebsmodus des DBL ein. Diese Angabe ist erforderlich, damit Binlademodule verarbeitet werden können.
- (8) Legt fest, dass aus den mit BLSLIB zugewiesenen Bibliotheken Module dynamisch nachgeladen werden. Diese Angabe ist erforderlich, wenn Sie das COBOL-Laufzeitsystem dynamisch nachladen möchten.

---

## 7 ESQL-COBOL-Anwendung unter openUTM

Dieses Kapitel beschreibt, was Sie beachten müssen, wenn Sie eine ESQL-COBOL-Anwendung als Teilprogramm einer UTM-Anwendung laufen lassen wollen.

Ausführliche Informationen zu UTM-Anwendungen finden Sie in den UTM Handbüchern „Anwendungen generieren und betreiben“ [11] und „Anwendungen programmieren“ [12].

### 7.1 Sprachumfang unter openUTM

In ESQL-COBOL-Anwendungen für den UTM-Betrieb sind zur Transaktionssicherung nur UTM-Sprachmittel erlaubt. Die SQL-Anweisungen COMMIT WORK und ROLLBACK WORK zum Beenden und Zurücksetzen von Transaktionen dürfen Sie daher nicht verwenden. Mit der Precompiler-Option SOURCE-PROPERTIES können Sie bereits bei der Vorübersetzung den verwendeten Sprachumfang prüfen (siehe [Abschnitt „Eigenschaften des ESQL-COBOL-Programms festlegen“ auf Seite 98](#)).

UTM übernimmt die Synchronisation von UTM- und Datenbank-Transaktionen. Beim Beenden einer UTM-Transaktion beendet UTM automatisch auch die Datenbank-Transaktion.

## 7.2 UTM-Anwendung generieren

Dieser Abschnitt beschreibt lediglich die Besonderheiten, die Sie beim Generieren von UTM-Anwendungen mit ESQL-COBOL-Teilprogrammen beachten müssen. Die Generierung ist ausführlich in den UTM Handbüchern „[Anwendungen generieren und betreiben](#)“ [11] und „[Anwendungen programmieren](#)“ [12] beschrieben.

### KDCDEF-Anweisung DATABASE

Bei der KDCDEF-Anweisung DATABASE müssen Sie SESSQL als ENTRY-Namen für SESAM/SQL angeben. Es empfiehlt sich, das UTM-Verbindungsmodul SESUTMC dynamisch nachzuladen. Dazu geben Sie bei der DATABASE-Anweisung mit dem Parameter LIB die SESAM/SQL-Modulbibliothek an. Geben Sie die DATABASE-Anweisung an, wie folgt:

```
DATABASE TYPE=SESAM,ENTRY=SESSQL,LIB=sesam-modlib
```

Enthält die Anwendung auch CALL-DML-Teilprogramme, geben Sie zusätzlich folgende DATABASE-Anweisung an:

```
DATABASE TYPE=SESAM,ENTRY=SESAM,LIB=sesam-modlib
```

### Anschlussprogramm KDCROOT übersetzen

Vor der Übersetzung des Anschlussprogramms KDCROOT weisen Sie die SESAM-Makrobibliothek zu. Die Zuweisung ist vom verwendeten Assembler abhängig.

#### *Beispiel 1*

SESAM-Makrobibliothek zuweisen mit ASSEMH:

```
/SET-FILE-LINK LINK-NAME=UTMLIB,FILE-NAME=utm_makrobibliothek
/SET-FILE-LINK LINK-NAME=SESAMLIB,FILE=sesam_makrobibliothek
/START-PROGRAM FROM-FILE=$ASSEMBH
//COMPILE -
//SOURCE=kdcroot
//,MACRO-LIBRARY=(*LINK(LINK-NAME=SESAMLIB) -
.
.
```

*Beispiel 2*

SESAM-Makrobibliothek zuweisen mit ASSGEN:

```
/FILE sesam-makrobibliothek, LINK=ALTLIB2  
/EXEC $ASSGEN  
*COMOPT ALTLIB2  
.  
.
```

**Austausch von UTM-Programmen**

Die innerhalb eines UTM-Vorgangs verwendeten SQL-Bindeladmodule müssen eindeutige Einsprungnamen haben.

Beim Generieren von UTM-Anwendungen mit ESQL-COBOL-Teilprogrammen müssen Sie:

- neue Einsprungnamen verwenden oder
- die Eindeutigkeit innerhalb eines UTM-Vorgangs gewährleisten. Dazu müssen Sie alle UTM-Vorgänge beenden, die einen wiederbenutzten Einsprungnamen verwendet haben.

## 7.3 UTM-Anwendung starten

Vor dem Starten einer UTM-Anwendung mit ESQL-COBOL-Teilprogrammen müssen Sie die SESAM/SQL-Modulbibliothek zuweisen. Es empfiehlt sich, benötigte Laufzeitsysteme beim Starten der UTM-Anwendung dynamisch nachzuladen.

In einer UTM-Anwendung können die Parameter für das SESAM/SQL-Konnektionsmodul über eine (globale) Konfigurationsdatei angegeben werden oder auch innerhalb der UTM-Startprozedur. Die Angaben in der Konfigurationsdatei haben Vorrang vor den UTM-Startparametern. Diese werden bei Angabe einer Konfigurationsdatei ignoriert.

Eine Beschreibung der Konfigurationsdatei und der Startparameter finden Sie im „[Basis-handbuch](#)“ [1].

Erweitern Sie die Startprozedur der UTM-Anwendung um folgende Anweisungen:

```

/SET-FILE-LINK LINK-NAME=SESAMOML, FILE-NAME=sesam-modlib (1)
.
.
/CONNECT-SESAM-CONFIGURATION TO-FILE=globale konfigurationsdatei, -
/ CONFIGURATION-LINK=linkname
oder
/SET-FILE-LINK LINK-NAME=SESCONF, FILE-NAME=dateiname (2)
/START-PROGRAM FROM-FILE=*MODULE(LIBRARY=benutzerbibl, ELEMENT=elementname - (3)
/ ,RUN-MODE=ADVANCED -
/ (ALTERNATE-LIBRARIES =YES) -
/ )
.UTM utm-startparameter (4)
.FHS fhs-startparameter (5)
.UTM END (6)
.
.
```

- (1) SESAM/SQL-Modulbibliothek zuweisen.
- (2) Konfigurationsdatei zuweisen.
- (3) Dynamischen Bindelader DBL zum Starten der UTM-Anwendung aufrufen.
- (4) UTM-Startparameter angeben. Eine Beschreibung der einzelnen Startparameter finden Sie im UTM Handbuch „[Anwendungen generieren und betreiben](#)“ [11].
- (5) Startparameter für das Formatierungssystem FHS angeben. Eine Beschreibung der einzelnen Startparameter finden Sie im Handbuch „[FHS \(TRANSDATA\)](#)“ [14].
- (6) Eingabe der UTM-Startparameter beenden.

---

## 8 Beispielprogramme

Die folgenden Beispielprogramme zeigen Anwendungsmöglichkeiten von ESQL-COBOL. Alle Beispielprogramme beziehen sich auf die Beispieldatenbank:

- Daten satzweise ausgeben, siehe „[Programm ABFRAGE](#)“ auf Seite 120
- Daten ändern, siehe „[Programm AENDERN](#)“ auf Seite 125
- Satz einfügen, siehe „[Programm EINFUEGEN](#)“ auf Seite 128
- Satz löschen, siehe „[Programm LOESCHEN](#)“ auf Seite 131
- Dynamische SQL-Anweisung angeben, siehe „[Programm DYNAMISCH](#)“ auf Seite 135

Nach jeder SQL-Anweisung wird geprüft, ob die SQL-Anweisung fehlerfrei durchgeführt wurde. Ist ein Fehler aufgetreten, enthält SQLSTATE einen entsprechenden Rückgabewert. Im Fehlerfall werden der Rückgabewert und die Zeilennummer des vom ESQL-Pre-compiler erzeugten COBOL-Programms ausgegeben. Die in den Beispielen definierten Indikatorvariablen werden aus Gründen der besseren Übersichtlichkeit nicht immer ausgewertet.

## 8.1 Programm ABFRAGE

Im Programm ABFRAGE werden aus der Tabelle AUFTRAG die von einer Firma vergebenen Aufträge ausgegeben. Der Benutzer kann angeben, von welcher Firma die Aufträge gezeigt werden sollen. Im Programm wird ein Cursor definiert. Die Daten der Cursortabelle werden satzweise ausgegeben. Mit der Precompiler-Option SOURCE-PROPERTIES sind die voreingestellte Datenbank und das voreingestellte Schema festgelegt (CATALOG=BEISPIEL, SCHEMA=AUFTRAGSVER).

```
*=====
IDENTIFICATION DIVISION.
PROGRAM-ID.    ABFRAGE.
*=====

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    TERMINAL IS DSG.
INPUT-OUTPUT SECTION.
*=====

DATA DIVISION.
*=====
*
WORKING-STORAGE SECTION.
*
*** AUSGABEVARIABLEN FUER AUFTRAG.
*
01 CAAUSGABE.
    02 CAANR          PIC ZZZ9.
    02 CAADATUMJ     PIC Z9999.
    02 FILLER        PIC X VALUE "-".
    02 CAADATUMM     PIC 99.
    02 FILLER        PIC X VALUE "-".
    02 CAADATUMT     PIC 99.
    02 FILLER        PIC X VALUE " ".
    02 CAATEXT       PIC X(30).
    02 CAASTAT       PIC Z9.
*** AUSGABEVARIABLEN FUER FEHLERAUSGABE.
*
01 AUSGABE.
    02 FILLER          PIC X(7) VALUE "FEHLER ".
    02 ASQLSTATE       PIC X(5).
    02 FILLER          PIC X(10) VALUE " IN ZEILE ".
    02 ASQLLINE       PIC Z(8)9.
*** KOMMUNIKATIONSBEREICH EINFUEGEN.
*
EXEC SQL
    INCLUDE SQLCA
```

```
        END-EXEC
***  BENUTZERVARIABLEN.
*
        EXEC SQL BEGIN DECLARE SECTION END-EXEC
***  SQLSTATE.
*
        01 SQLSTATE      PIC X(5).
***  FELD ZUR AUFNAHME DES EINGEGEBENEN FIRMENNAMENS.
*
        01 FIRMENNAME    PIC X(40).
***  FELD ZUR AUFNAHME DER HERAUSGESUCHTEN KUNDENNUMMER.
*
        01 KUNDENNR      PIC S9(9) USAGE IS BINARY.
***  BENUTZERVARIABLEN FUER AUFTRAG.
*
        01 AUFTRAG.
          02 ANR          PIC S9(9) USAGE IS BINARY.
          02 ADATUM DATE.
              03 JAHR     PIC S9(4) USAGE IS BINARY.
              03 MONAT    PIC S9(4) USAGE IS BINARY.
              03 TAG       PIC S9(4) USAGE IS BINARY.
          02 ATEXT        PIC X(30).
          02 ASTAT        PIC S9(9) USAGE IS BINARY.
***  INDIKATORVARIABLEN FUER AUFTRAG.
*
        01 INDAUFTRAG.
          02 INDANR        PIC S9(4) USAGE IS BINARY.
          02 INDADATUM     PIC S9(4) USAGE IS BINARY.
          02 INDATEXT      PIC S9(4) USAGE IS BINARY.
          02 INDASTAT      PIC S9(4) USAGE IS BINARY.
***
*
        EXEC SQL END DECLARE SECTION END-EXEC
*=====
        PROCEDURE DIVISION.
        STEUER SECTION.
        S-0.
***  FEHLERBEHANDLUNG.
*
        EXEC SQL
            WHENEVER SQLERROR GOTO NACHBEREITEN
        END-EXEC
        S-1.
        PERFORM ANFANG.
        S-2.
        PERFORM ABFRAGE.
        S-3.
        PERFORM ENDE.
```

```

S-9.
*****PROGRAMMENDE*****
      STOP RUN.
*=====
ANFANG SECTION.
*=====
A-1.
      DISPLAY "*** PROGRAMMANFANG ***" UPON DSG.
A-9.
      EXIT.
*=====
ABFRAGE SECTION.
*=====
F-1.
      DISPLAY "GEBEN SIE DEN NAMEN DER FIRMA EIN," UPON DSG.
      DISPLAY "DEREN AUFTRAEGE SIE ANZEIGEN MOECHTEN." UPON DSG.
      ACCEPT FIRMENNAME FROM DSG.
*** ENDE, WENN KUNDE NICHT GEFUNDEN
*
      EXEC SQL
          WHENEVER NOT FOUND GOTO KNRENDE
      END-EXEC
*** SATZ AUS DER TABELLE "KUNDE" WAEHLEN.
*
      EXEC SQL
          SELECT KNR
             INTO :KUNDENNR
             FROM KUNDE
             WHERE FIRMA = :FIRMENNAME
      END-EXEC
      DISPLAY "AUFTRAEGE DES KUNDEN ", FIRMENNAME UPON DSG.
*** CURSOR FUER DIE AUFTRAEGE DES ANGEGEBENEN KUNDEN
*** DEFINIEREN.
*
      EXEC SQL
          DECLARE CUR_AUFTRAG CURSOR FOR
              SELECT ANR, ADATUM, ATEXT, ASTAT
                 FROM AUFTRAG
                 WHERE KNR = :KUNDENNR
      END-EXEC
*** CURSOR OEFFNEN.
*
      EXEC SQL
          OPEN CUR_AUFTRAG
      END-EXEC.
*** SCHLEIFENENDE, WENN TABELLENENDE ERREICHT.
*
      EXEC SQL

```

```
        WHENEVER NOT FOUND GOTO TABENDE
    END-EXEC
*** CURSOR AUF DEN ERSTEN BZW. NAECHSTEN SATZ POSITIONIEREN.
*
F-2.
    EXEC SQL
        FETCH CUR_AUFTRAG
            INTO :ANR      INDICATOR :INDANR,
                :ADATUM  INDICATOR :INDADATUM,
                :ATEXT   INDICATOR :INDATEXT,
                :ASTAT   INDICATOR :INDASTAT
    END-EXEC.
F-3.
    MOVE ANR OF AUFTRAG TO CAANR
    IF INDADATUM = -1
        THEN
            MOVE -1 TO CAADATUMJ
            MOVE 0 TO CAADATUMM
            MOVE 0 TO CAADATUMT
        ELSE
            MOVE JAHR OF ADATUM TO CAADATUMJ
            MOVE MONAT OF ADATUM TO CAADATUMM
            MOVE TAG OF ADATUM TO CAADATUMT
        END-IF
    IF INDATEXT = -1
        THEN
            MOVE "NULL" TO CAATEXT
        ELSE
            MOVE ATEXT TO CAATEXT
        END-IF
    MOVE ASTAT TO CAASTAT
    DISPLAY CAAUSGABE UPON DSG
    GO TO F-2.
TABENDE.
*** TABELLENENDE ERREICHT
*
    EXEC SQL
        WHENEVER NOT FOUND CONTINUE
    END-EXEC
    DISPLAY "TABELLENENDE ERREICHT" UPON DSG
*** CURSOR SCHLIESSEN.
*
    EXEC SQL
        CLOSE CUR_AUFTRAG
    END-EXEC.
    GO TO F-9.
KNRENDE.
*** KUNDE NICHT GEFUNDEN
```

```

*
    EXEC SQL
        WHENEVER NOT FOUND CONTINUE
    END-EXEC
    DISPLAY "KUNDE ", FIRMENNAME, " NICHT GEFUNDEN" UPON DSG.
F-9.
    EXIT.
*=====
    ENDE SECTION.
*=====
    E-1.
*** TRANSAKTION BEENDEN.
*
    EXEC SQL
        COMMIT WORK
    END-EXEC.
E-2.
    DISPLAY "*** ORDNUNGSGEMAESSES PROGRAMMENDE ***" UPON DSG.
E-9.
    EXIT.
*=====
    NACHBEREITEN SECTION.
*=====
    N-1.
    EXEC SQL
        WHENEVER SQLERROR CONTINUE
    END-EXEC
    MOVE SQLSTATE TO ASQLSTATE
    MOVE SQLLINE TO ASQLLINE
    DISPLAY AUSGABE UPON DSG
*** TRANSAKTION ZURUECKSETZEN.
*
    EXEC SQL
        ROLLBACK WORK
    END-EXEC.
N-2.
    DISPLAY "*** DAS PROGRAMM KONNTE NICHT VOLLSTAENDIG"
-      "DURCHLAUFEN WERDEN ***" UPON DSG
    STOP RUN.
N-9.
    EXIT.

```

## 8.2 Programm AENDERN

Im Programm AENDERN werden in der Tabelle FARBTAB die Farbanteile einer Farbe geändert. Mit der Precompiler-Option SOURCE-PROPERTIES sind die voreingestellte Datenbank und das voreingestellte Schema festgelegt (CATALOG=BEISPIEL, SCHEMA=TEILE).

```
*=====
IDENTIFICATION DIVISION.
PROGRAM-ID.    AENDERN.
*=====

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    TERMINAL IS DSG.
INPUT-OUTPUT SECTION.
*=====

DATA DIVISION.
*=====

WORKING-STORAGE SECTION.
*** AUSGABEVARIABLEN FUER FEHLERAUSGABE.
*
01 AUSGABE.
    02 FILLER                PIC X(7) VALUE "FEHLER ".
    02 ASQLSTATE             PIC X(5).
    02 FILLER                PIC X(10) VALUE " IN ZEILE ".
    02 ASQLLINE              PIC Z(8)9.
*** AUSGABEVARIABLEN.
*
01 FAUSGABE.
    02 FRED                  PIC Z9.99.
    02 FGREEN                PIC Z9.99.
    02 FBLUE                 PIC Z9.99.
*** KOMMUNIKATIONSBEREICH EINFUEGEN.
*
    EXEC SQL
        INCLUDE SQLCA
    END-EXEC
*** BENUTZERVARIABLEN.
*
    EXEC SQL BEGIN DECLARE SECTION END-EXEC
*** SQLSTATE.
*
01 SQLSTATE                PIC X(5).
*** VEKTOR FUER RGB-ANTEILE EINER FARBE.
*
01 FARBE.
```

```

        02 FARBANTEILE PIC SV99 OCCURS 3 TIMES.
        EXEC SQL END DECLARE SECTION END-EXEC
*=====
PROCEDURE DIVISION.
    STEUER SECTION.
    S-0.
*** FEHLERBEHANDLUNG.
*
        EXEC SQL
            WHENEVER SQLERROR GOTO NACHBEREITEN
        END-EXEC
    S-1.
        PERFORM ANFANG.
    S-2.
        PERFORM AENDERN.
    S-3.
        PERFORM ENDE.
    S-9.
*****PROGRAMMENDE*****
        STOP RUN.
*=====
ANFANG SECTION.
*=====
    A-1.
        DISPLAY "*** PROGRAMMANFANG ***" UPON DSG.
    A-9.
        EXIT.
*=====
AENDERN SECTION.
*=====
    U-1.
        EXEC SQL
            SELECT RGB(1..3) INTO :FARBANTEILE(1..3)
                FROM FARBTAB
                WHERE FARBNAME = 'himmelblau'
        END-EXEC
        MOVE FARBANTEILE(1) TO FRED
        MOVE FARBANTEILE(2) TO FGREEN
        MOVE FARBANTEILE(3) TO FBLUE
        DISPLAY "Farbanteile fuer himmelblau: ", FAUSGABE
            UPON DSG.
    U-2.
        MOVE 0.1 TO FARBANTEILE(1)
        MOVE 0.2 TO FARBANTEILE(2)
*** SATZ IN DER TABELLE "FARBTAB" AENDERN.
*
        EXEC SQL
            UPDATE FARBTAB

```

```

                SET RGB(1..2) = :FARBANTEILE(1..2)
                WHERE FARBNAME = 'himmelblau'
    END-EXEC.
U-3.
    EXEC SQL
        SELECT RGB(1..3) INTO :FARBANTEILE(1..3)
            FROM FARBTAB
            WHERE FARBNAME = 'himmelblau'
    END-EXEC
    MOVE FARBANTEILE(1) TO FRED
    MOVE FARBANTEILE(2) TO FGREEN
    MOVE FARBANTEILE(3) TO FBLUE
    DISPLAY "Farbanteile fuer himmelblau: ", FAUSGABE
        UPON DSG.
U-9.
    EXIT.
*=====
    ENDE SECTION.
*=====
E-1.
*** TRANSAKTION BEENDEN.
    EXEC SQL
        COMMIT WORK
    END-EXEC.
E-2.
    DISPLAY "*** ORDNUNGSGEMAESSES PROGRAMMENDE ***" UPON DSG.
E-9.
    EXIT.
*=====
    NACHBEREITEN SECTION.
*=====
N-1.
    EXEC SQL
        WHENEVER SQLERROR CONTINUE
    END-EXEC
    MOVE SQLSTATE TO ASQLSTATE
    MOVE SQLLINE TO ASQLLINE
    DISPLAY AUSGABE UPON DSG
*** TRANSAKTION ZURUECKSETZEN.
    EXEC SQL
        ROLLBACK WORK
    END-EXEC.
N-2.
    DISPLAY "*** DAS PROGRAMM KONNTE NICHT VOLLSTAENDIG DURCHLAUFE"
-         "N WERDEN ***" UPON DSG
    STOP RUN.
N-9.
    EXIT.

```

## 8.3 Programm EINFUEGEN

Im Programm EINFUEGEN wird in der Tabelle AUFTRAG ein Satz eingefügt. Mit der Pre-compiler-Option SOURCE-PROPERTIES sind die voreingestellte Datenbank und das voreingestellte Schema festgelegt (CATALOG=BEISPIEL, SCHEMA=AUFTRAGSVER).

```

*=====
IDENTIFICATION DIVISION.
PROGRAM-ID.    EINFUEGEN.
*=====

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    TERMINAL IS DSG.
INPUT-OUTPUT SECTION.
*=====

DATA DIVISION.
*=====

WORKING-STORAGE SECTION.
*** AUSGABEVARIABLEN FUER FEHLERAUSGABE.
*
01 AUSGABE.
    02 FILLER            PIC X(7) VALUE "FEHLER ".
    02 ASQLSTATE         PIC X(5).
    02 FILLER            PIC X(10) VALUE " IN ZEILE ".
    02 ASQLLINE          PIC Z(8)9.
*** KOMMUNIKATIONSBEREICH EINFUEGEN.
*
    EXEC SQL
        INCLUDE SQLCA
    END-EXEC
*** BENUTZERVARIABLEN.
*
    EXEC SQL BEGIN DECLARE SECTION END-EXEC
*** SQLSTATE.
*
01 SQLSTATE      PIC X(5).
*** BENUTZERVARIABLEN FUER AUFTRAG.
*
01 AUFTRAG.
    02 ANR        PIC S9(9) USAGE IS BINARY.
    02 KNR        PIC S9(9) USAGE IS BINARY.
    02 KONR       PIC S9(9) USAGE IS BINARY.
    02 ATEXT      PIC X(30).
    02 FERTIGSOLL DATE.
        03 JAHR   PIC S9(4) USAGE IS BINARY.
        03 MONAT  PIC S9(4) USAGE IS BINARY.

```

```

          03 TAG      PIC S9(4) USAGE IS BINARY.
          02 ASTAT   PIC S9(9) USAGE IS BINARY.
***
      EXEC SQL END DECLARE SECTION END-EXEC
*=====
      PROCEDURE DIVISION.
      STEUER SECTION.
      S-0.
*** FEHLERBEHANDLUNG.
*
      EXEC SQL
          WHENEVER SQLERROR GOTO NACHBEREITEN
      END-EXEC
      S-1.
      PERFORM ANFANG.
      S-2.
      PERFORM EINFUEGEN.
      S-3.
      PERFORM ENDE.
      S-9.
*****PROGRAMMENDE*****
      STOP RUN.
*=====
      ANFANG SECTION.
*=====
      A-1.
          DISPLAY "*** PROGRAMMANFANG ***" UPON DSG.
      A-9.
          EXIT.
*=====
      EINFUEGEN SECTION.
*=====
      I-1.
          MOVE 345 TO ANR
          MOVE 101 TO KNR
          MOVE 20 TO KONR
          MOVE "Netzwerkinstallation" TO ATEXT
          MOVE 1991 TO JAHR OF FERTIGSOLL
          MOVE 6 TO MONAT OF FERTIGSOLL
          MOVE 20 TO TAG OF FERTIGSOLL
          MOVE 1 TO ASTAT
*** SATZ IN DIE TABELLE "AUFTRAG" EINFUEGEN.
*
      EXEC SQL
          INSERT INTO AUFTRAG (ANR, KNR, KONR, ATEXT,
                              FERTIGSOLL, ASTAT)
          VALUES (:ANR, :KNR, :KONR, :ATEXT,
                  :FERTIGSOLL, :ASTAT)

```

```

        END-EXEC.
I-9.
    EXIT.
*=====
    ENDE SECTION.
*=====
    E-1.
*** TRANSAKTION BEENDEN.
*
    EXEC SQL
        COMMIT WORK
    END-EXEC.
    E-2.
    DISPLAY "*** ORDNUNGSGEMAESSES PROGRAMMENDE ***" UPON DSG.
    E-9.
    EXIT.
*=====
    NACHBEREITEN SECTION.
*=====
    N-1.
    EXEC SQL
        WHENEVER SQLERROR CONTINUE
    END-EXEC
    MOVE SQLSTATE TO ASQLSTATE
    MOVE SQLLINE TO ASQLLINE
    DISPLAY AUSGABE UPON DSG
*** TRANSAKTION ZURUECKSETZEN.
*
    EXEC SQL
        ROLLBACK WORK
    END-EXEC.
    N-2.
    DISPLAY "*** DAS PROGRAMM KONNTE NICHT VOLLSTAENDIG DURCHLAUFE
-         "N WERDEN ***" UPON DSG
    STOP RUN.
    N-9.
    EXIT.

```

## 8.4 Programm LOESCHEN

Im Programm LOESCHEN werden alle Aufträge gelöscht, die bereits abgelegt sind (Aufträge mit Auftragsstatus 5). Es werden aus der Tabelle LEISTUNG alle Angaben zur Leistung gelöscht. Dann wird der Auftrag auch in der Tabelle AUFTRAG gelöscht. Mit der Precompiler-Option SOURCE-PROPERTIES sind die voreingestellte Datenbank und das voreingestellte Schema festgelegt (CATALOG=BEISPIEL, SCHEMA=AUFTRAGSVER).

```
*=====
IDENTIFICATION DIVISION.
PROGRAM-ID.    LOESCHEN.
*=====

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    TERMINAL IS DSG.
INPUT-OUTPUT SECTION.
*=====

DATA DIVISION.
*=====

WORKING-STORAGE SECTION.
*** AUSGABEVARIABLEN FUER FEHLERAUSGABE.
*
01 AUSGABE.
    02 FILLER            PIC X(7) VALUE "FEHLER ".
    02 ASQLSTATE         PIC X(5).
    02 FILLER            PIC X(10) VALUE " IN ZEILE ".
    02 ASQLLINE          PIC Z(8)9.
*** VARIABLE FUER DIE AUSGABE DER AUFTRAGSNUMMER.
*
01 AANR          PIC Z(8)9.
*** KOMMUNIKATIONSBEREICH EINFUEGEN.
*
    EXEC SQL
        INCLUDE SQLCA
    END-EXEC
*** BENUTZERVARIABLEN.
*
    EXEC SQL BEGIN DECLARE SECTION END-EXEC
*** SQLSTATE.
*
01 SQLSTATE     PIC X(5).
*** BENUTZERVARIABLEN FUER AUFTRAG.
*
01 AUFTRAG.
    02 ANR       PIC S9(9) USAGE IS BINARY.
    02 ATEXT     PIC X(30).
```

```

*** INDIKATORVARIABLEN FUER AUFTRAG.
*
01 INDAUFTRAG.
    02 INDANR          PIC S9(4) USAGE IS BINARY.
    02 INDATEXT       PIC S9(4) USAGE IS BINARY.
***
    EXEC SQL END DECLARE SECTION END-EXEC
*=====
PROCEDURE DIVISION.
STEUER SECTION.
S-0.
*** FEHLERBEHANDLUNG.
*
    EXEC SQL
        WHENEVER SQLERROR GOTO NACHBEREITEN
    END-EXEC
S-1.
    PERFORM ANFANG.
S-2.
    PERFORM LOESCHEN.
S-3.
    PERFORM ENDE.
S-9.
*****PROGRAMMENDE*****
    STOP RUN.
*=====
ANFANG SECTION.
*=====
A-1.
    DISPLAY "*** PROGRAMMANFANG ***" UPON DSG.
A-9.
    EXIT.
*=====
LOESCHEN SECTION.
*=====
D-1.
*** CURSOR FUER DIE BEREITS
*** ABGELEGTEN AUFTRAEGE DEFINIEREN (ASTAT=5).
*
    EXEC SQL
        DECLARE CUR_AUFTRAG CURSOR FOR
            SELECT ANR, ATEXT
            FROM AUFTRAG
            WHERE ASTAT = 5
    END-EXEC

```

```
*** CURSOR OEFFNEN.
*
    EXEC SQL
        OPEN CUR_AUFTRAG
    END-EXEC.
*** SCHLEIFENENDE, WENN TABELLENENDE ERREICHT.
*
    EXEC SQL
        WHENEVER NOT FOUND GOTO TABENDE
    END-EXEC.
*** CURSOR AUF DEN ERSTEN BZW. NAECHSTEN SATZ POSITIONIEREN.
*
D-2.
    EXEC SQL
        FETCH CUR_AUFTRAG
            INTO :ANR    INDICATOR :INDANR,
                :ATEXT  INDICATOR :INDATEXT
    END-EXEC.
D-3.
    MOVE ANR TO AANR
    DISPLAY "LOESCHEN DES AUFTRAGS ", AANR, ": ", ATEXT
        UPON DSG
*** ALLE ZUM AUFTRAG GEHOERENDEN ANGABEN ZUR LEISTUNG
*** IN DER TABELLE LEISTUNG LOESCHEN.
*
    EXEC SQL
        DELETE FROM LEISTUNG
            WHERE ANR = :ANR
    END-EXEC
*** AUFTRAG AUS DER TABELLE AUFTRAG LOESCHEN.
*
    EXEC SQL
        DELETE FROM AUFTRAG
            WHERE ANR = :ANR
    END-EXEC
    GO TO D-2.
TABENDE.
    DISPLAY "ALLE AUFTRAEGE, DIE BEREITS " UPON DSG.
    DISPLAY "ABGELEGT SIND," UPON DSG.
    DISPLAY "WURDEN GELOESCHT" UPON DSG.
*** CURSOR SCHLIESSEN.
*
    EXEC SQL
        CLOSE CUR_AUFTRAG
    END-EXEC.
D-9.
    EXIT.
```

```

*=====
ENDE SECTION.
*=====
E-1.
*** TRANSAKTION BEENDEN.
*
    EXEC SQL
        COMMIT WORK
    END-EXEC.
E-2.
    DISPLAY "*** ORDNUNGSGEMAESSES PROGRAMMENDE ***" UPON DSG.
E-9.
    EXIT.
*=====
NACHBEREITEN SECTION.
*=====
N-1.
    EXEC SQL
        WHENEVER SQLERROR CONTINUE
    END-EXEC
    MOVE SQLSTATE TO ASQLSTATE
    MOVE SQLLINE TO ASQLLINE
    DISPLAY AUSGABE UPON DSG
*** TRANSAKTION ZURUECKSETZEN.
*
    EXEC SQL
        ROLLBACK WORK
    END-EXEC.
N-2.
    DISPLAY "*** DAS PROGRAMM KONNTE NICHT VOLLSTAENDIG DURCHLAUFE
-         "N WERDEN ***" UPON DSG
    STOP RUN.
N-9.
    EXIT.

```

## 8.5 Programm DYNAMISCH

Das Programm DYNAMISCH ist ein Anwendungsbeispiel für dynamische SQL-Anweisungen. Das Programm gibt aus der Tabelle LEISTUNG Daten zu einem Auftrag aus. Die Benutzerin oder der Benutzer kann während des Programmablaufs eine Auftragsnummer und eine Suchbedingung für die zu zeigenden Daten eingeben. Im Programm wird ein dynamischer Cursor definiert. Die eingegebene Formulierung wird in eine Cursorbeschreibung umgesetzt. Die Cursorbeschreibung wird dynamisch übersetzt und der zugeordnete Cursor wird geöffnet. Anschließend kann der Benutzer eine dynamische SQL-Anweisung eingeben, um in der Tabelle LEISTUNG Daten zu ändern, löschen oder einzufügen. Die dynamische SQL-Anweisung wird in einem Schritt vorbereitet und ausgeführt. Mit der Precompiler-Option SOURCE-PROPERTIES sind die voreingestellte Datenbank und das voreingestellte Schema festgelegt (CATALOG=BEISPIEL, SCHEMA=AUFTRAGSVER).

```
*=====
IDENTIFICATION DIVISION.
PROGRAM-ID.    DYNAMISCH.
*=====
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    TERMINAL IS DSG.
INPUT-OUTPUT SECTION.
*=====
DATA DIVISION.
*=====
WORKING-STORAGE SECTION.
*** AUSGABEVARIABLEN FUER FEHLERAUSGABE.
*
01 AUSGABE.
    02 FILLER                PIC X(7) VALUE "FEHLER ".
    02 ASQLSTATE             PIC X(5).
    02 FILLER                PIC X(10) VALUE " IN ZEILE ".
    02 ASQLLINE              PIC Z(8)9.
*** VARIABLE ZUR AUSGABE EINES SATZES DER TABELLE LEISTUNG.
*
01 LAUSGABE.
    02 LALNR                 PIC ZZZZ9.
    02 LAANR                 PIC ZZZZ9.
    02 LALDATUM.
        03 JAHR              PIC Z9999.
        03 FILLER            PIC X          VALUE "-".
        03 MONAT             PIC 99.
        03 FILLER            PIC X          VALUE "-".
        03 TAG                PIC 99.
        03 FILLER            PIC X          VALUE " ".
    02 LALTEXT                PIC X(25).
```

```

    02 FILLER          PIC X          VALUE " ".
    02 LALEINHEIT     PIC X(10).
    02 LALANZ         PIC ZZ9.99.
    02 LALSATZ        PIC ZZZZ9.
    02 LAMWSATZ       PIC Z9.99.
    02 LARNR          PIC ZZZ9.
*** FELD ZUR AUFNAHME DER EINGEGEBEN AUFTRAGSNUMMER
*
    01 ANREINGABE     PIC Z(9).
*** FELD ZUR AUFNAHME DER BEDINGUNG FUER DIE SUCHE
*
    01 BEDINGUNG      PIC X(200).
*** KOMMUNIKATIONSBEREICH EINFUEGEN.
*
    EXEC SQL
        INCLUDE SQLCA
    END-EXEC
*** BENUTZERVARIABLEN.
*
    EXEC SQL BEGIN DECLARE SECTION END-EXEC
*** SQLSTATE.
*
    01 SQLSTATE       PIC X(5).
*** FELD ZUR AUFNAHME DER EINGEGEBEN AUFTRAGSNUMMER.
*
    01 AUFTRAGSNR     PIC S9(9) USAGE IS BINARY.
*** FELD ZUR AUFNAHME DER CURSORBESCHREIBUNG.
*
    01 BESCHREIBUNG  PIC X(200).
*** FELD ZUR AUFNAHME DER DYNAMISCHEN ANWEISUNG.
*
    01 ANWEISUNG      PIC X(200).
*** BENUTZERVARIABLEN FUER LEISTUNG.
*
    01 LEISTUNG.
        02 LNR          PIC S9(9) USAGE IS BINARY.
        02 ANR          PIC S9(9) USAGE IS BINARY.
        02 LDATUM DATE.
            03 JAHR      PIC S9(4) USAGE IS BINARY.
            03 MONAT     PIC S9(4) USAGE IS BINARY.
            03 TAG       PIC S9(4) USAGE IS BINARY.
        02 LTEXT       PIC X(25).
        02 LEINHEIT    PIC X(10).
        02 LANZ        PIC S999V99.
        02 LSATZ       PIC S9(4).
        02 MWSATZ      PIC SV99.
        02 RNR         PIC S9(4).
*** INDIKATORVARIABLEN FUER LEISTUNG.

```

```

*
01 INDLEISTUNG.
   02 INDLNR          PIC S9(4) USAGE IS BINARY.
   02 INDANR          PIC S9(4) USAGE IS BINARY.
   02 INDLDATUM       PIC S9(4) USAGE IS BINARY.
   02 INDLTEXT        PIC S9(4) USAGE IS BINARY.
   02 INDLEINHEIT     PIC S9(4) USAGE IS BINARY.
   02 INDLANZ         PIC S9(4) USAGE IS BINARY.
   02 INDLSATZ        PIC S9(4) USAGE IS BINARY.
   02 INDMWSATZ       PIC S9(4) USAGE IS BINARY.
   02 INDRNR          PIC S9(4) USAGE IS BINARY.

***
      EXEC SQL END DECLARE SECTION END-EXEC
=====
PROCEDURE DIVISION.
STEUER SECTION.
S-1.
*** FEHLERBEHANDLUNG.
*
      EXEC SQL
            WHENEVER SQLERROR GOTO NACHBEREITEN
      END-EXEC

S-2.
      PERFORM ANFANG.

S-3.
      PERFORM CURSOR-ZUGRIFF.

S-4.
      PERFORM DYNAMISCH.

S-5.
      PERFORM ENDE.

S-9.
*****PROGRAMMENDE*****
      STOP RUN.
*=====
ANFANG SECTION.
*=====
A-1.
      DISPLAY "** PROGRAMMANFANG **" UPON DSG.

A-9.
      EXIT.
*=====
CURSOR-ZUGRIFF SECTION.
*=====
C-1.
      DISPLAY "GEBEN SIE DIE AUFTRAGSNUMMER EIN (NEUNSTELLIG)." UPON DSG.
      ACCEPT ANREINGABE FROM DSG

***
      DISPLAY "GEBEN SIE DIE BEDINGUNG EIN," UPON DSG.

```

```

        DISPLAY "DIE FUER DIE AUSZUGEBENDEN LEISTUNGEN GELTEN SOLL."
        UPON DSG
        ACCEPT BEDINGUNG FROM DSG.
C-2.
*** CURSORBESCHREIBUNG ERZEUGEN.
*
        INITIALIZE BESCHREIBUNG
        STRING "SELECT * FROM LEISTUNG WHERE ANR = ? AND " BEDINGUNG
            DELIMITED BY SIZE INTO BESCHREIBUNG
*** CURSORBESCHREIBUNG VORBEREITEN.
*
        EXEC SQL
            PREPARE CURBESCHREIBUNG FROM :BESCHREIBUNG
        END-EXEC.
C-3.
*** CURSOR FUER DIE LEISTUNGEN ZUM ANGEGEBENEN AUFTRAG
*** DEFINIEREN.
*
        EXEC SQL
            DECLARE CUR_LEISTUNG CURSOR FOR CURBESCHREIBUNG
        END-EXEC
*** CURSOR OEFFNEN.
*** DABEI WIRD DER PLATZHALTER IN DER CURSORBESCHREIBUNG
*** MIT DER GEWAELHTEN AUFTRAGSNUMMER BESETZT.
*
        MOVE ANREINGABE TO AUFTRAGSNR
        EXEC SQL
            OPEN CUR_LEISTUNG USING :AUFTRAGSNR
        END-EXEC.
*** SCHLEIFENENDE, WENN TABELLENENDE ERREICHT.
*
        EXEC SQL
            WHENEVER NOT FOUND GOTO TABENDE
        END-EXEC
*** CURSOR AUF DEN ERSTEN/NAECHSTEN SATZ POSITIONIEREN.
*
C-4.
        EXEC SQL
            FETCH CUR_LEISTUNG
                INTO :LNR          INDICATOR :INDLNR,
                   :ANR          INDICATOR :INDANR,
                   :LDATUM       INDICATOR :INDLDATUM,
                   :LTEXT        INDICATOR :INDLTEXT,
                   :LEINHEIT     INDICATOR :INDLEINHEIT,
                   :LANZ         INDICATOR :INDLANZ,
                   :LSATZ        INDICATOR :INDLSATZ,
                   :MWSATZ       INDICATOR :INDMWSATZ,
                   :RNR          INDICATOR :INDRNR

```

```
END-EXEC
MOVE LNR TO LALNR
MOVE ANR TO LAANR
MOVE JAHR OF LDATUM TO JAHR OF LALDATUM
MOVE MONAT OF LDATUM TO MONAT OF LALDATUM
MOVE TAG OF LDATUM TO TAG OF LALDATUM
MOVE LTEXT TO LALTEXT
MOVE LEINHEIT TO LALEINHEIT
MOVE LANZ TO LALANZ
MOVE LSATZ TO LALSATZ
MOVE MWSATZ TO LAMWSATZ
MOVE RNR TO LARNR
DISPLAY LAUSGABE UPON DSG
GO TO C-4.

TABENDE.
*** TABELLENENDE ERREICHT
*
EXEC SQL
    WHENEVER NOT FOUND CONTINUE
END-EXEC
DISPLAY "TABELLENENDE ERREICHT" UPON DSG
*** CURSOR SCHLIESSEN.
*
EXEC SQL
    CLOSE CUR_LEISTUNG
END-EXEC.

C-9.
EXIT.

*=====
DYNAMISCH SECTION.
*=====

D-1.
DISPLAY "GEBEN SIE EINE SQL-ANWEISUNG" UPON DSG
DISPLAY "(INSERT, UPDATE ODER DELETE) EIN. ABRUCH MIT '*' "
    UPON DSG
INITIALIZE ANWEISUNG
ACCEPT ANWEISUNG FROM DSG.

D-2.
IF ANWEISUNG NOT = "*"
    THEN
*** DYNAMISCHE ANWEISUNG
*** IN EINEM SCHRITT VORBEREITEN UND AUSFUEHREN.
*
EXEC SQL
    EXECUTE IMMEDIATE :ANWEISUNG
END-EXEC
DISPLAY "ANWEISUNG AUSGEFUEHRT" UPON DSG
END-IF.
```

```

D-9.
    EXIT.
*=====
ENDE SECTION.
*=====
E-1.
*** TRANSAKTION BEENDEN.
*
    EXEC SQL
        COMMIT WORK
    END-EXEC.
E-2.
    DISPLAY "*** ORDNUNGSGEMAESSES PROGRAMMENDE ***" UPON DSG.
E-9.
    EXIT.
*=====
NACHBEREITEN SECTION.
*=====
N-1.
    EXEC SQL
        WHENEVER SQLERROR CONTINUE
    END-EXEC
    MOVE SQLSTATE TO ASQLSTATE
    MOVE SQLLINE TO ASQLLINE
    DISPLAY AUSGABE UPON DSG
*** TRANSAKTION ZURUECKSETZEN.
*
    EXEC SQL
        ROLLBACK WORK
    END-EXEC.
N-2.
    DISPLAY "*** DAS PROGRAMM KONNTE NICHT VOLLSTAENDIG DURCHLAUFE
-         "N WERDEN ***" UPON DSG
    STOP RUN.
N-9.
    EXIT.

```

---

## 9 Meldungen des ESQL-COBOL-Systems

IQB0001 ESQL-COBOL V2.0A00  
IQB0001 ESQL-COBOL V2.0A00

IQB0101 (&00) INSUFFICIENT MEMORY (REASON (&01))  
IQB0101 (&00) SPEICHERMANGEL (GRUND (&01))

IQB0201 (&00) OPEN FAILURE ON FILE <(&01)> (REASON (&02))  
IQB0201 (&00) DATEI <(&01)> KANN NICHT GEOEFFNET WERDEN (GRUND (&02))

IQB0202 (&00) CLOSE FAILURE ON FILE <(&01)> (REASON (&02))  
IQB0202 (&00) DATEI <(&01)> KANN NICHT GESCHLOSSEN WERDEN (GRUND (&02))

IQB0203 (&00) WRITE FAILURE ON FILE <(&01)> (REASON (&02))  
IQB0203 (&00) SCHREIBFEHLER AUF DATEI <(&01)> (GRUND (&02))

IQB0204 (&00) READ FAILURE ON FILE <(&01)> (REASON (&02))  
IQB0204 (&00) LESEFEHLER AUF DATEI <(&01)> (GRUND (&02))

IQB0205 (&00) NO LIBRARY PATH DECLARED TO INCLUDE <(&01)>  
IQB0205 (&00) KEIN BIBLIOTHEKSPFAD ERKLAERT FUER INCLUDE <(&01)>

IQB0206 (&00) LIBRARY PATH FAILURE ON INCLUDE <(&01)> (REASON (&02))  
IQB0206 (&00) FEHLER IM BIBLIOTHEKSPFAD FUER INCLUDE <(&01)> (GRUND (&02))

IQB0207 (&00) NESTING LIMIT EXCEEDED ON INCLUDE OF <(&01)>  
IQB0207 (&00) MAX VERSCHACHTELUNGSTIEFE ERREICHT BEIM INCLUDIEREN VON <(&01)>

IQB0208 (&00) INPUT FILE <(&01)> RECORD SIZE EXCEEDS NORM, TRUNCATED  
IQB0208 (&00) EINGABEDATEI <(&01)>: SATZLAENGE ZU GROSS; SATZ GEKUERZT

IQB0209 (&00) OUTPUT FILE <(&01)> RECORD SIZE <(&02)> EXCEEDS NORM, TRUNCATED  
IQB0209 (&00) AUSGABEDATEI <(&01)>: SATZLAENGE <(&02)> ZU GROSS; SATZ GEKUERZT

IQB0220 (&00) UNRECOVERABLE I/O ERROR OCCURRED  
IQB0220 (&00) SCHWERER EIN-/AUSGABEFehler

IQB0301 (&00) SDF: ERROR CODE (0X(&01))  
IQB0301 (&00) SDF: FEHLER CODE (0X(&01))

IQB0302 (&00) SDF: ILLEGAL STATEMENT (0X(&01))  
IQB0302 (&00) SDF: ANWEISUNG (0X(&01)) NICHT ERLAUBT

IQB0303 (&00) SDF: UNRECOVERABLE SYSTEM ERROR  
IQB0303 (&00) SDF: SCHWERER SYSTEM-Fehler

IQB0304 (&00) SDF: OPERAND ERROR DETECTED  
IQB0304 (&00) SDF: OPERANDEN-FEHLER

IQB0305 (&00) SDF: TRANSFER AREA IS TOO SMALL  
IQB0305 (&00) SDF: UEBERGABEBEREICH ZU KLEIN

IQB0306 (&00) SDF: INCORRECT STATEMENT PROCESSED  
IQB0306 (&00) SDF: FALSCHER ANWEISUNG VERARBEITET

IQB0307 (&00) SDF: INCORRECT STATEMENT (END/STEP PROCESSED)  
IQB0307 (&00) SDF: FALSCHER ANWEISUNG (END/STEP DURCHGEFUEHRT)

IQB0308 (&00) SDF: ERRONEOUS DEFAULT VALUES IN STATEMENT SYNTAX  
IQB0308 (&00) SDF: FALSCHER DEFAULT-WERT IN ANWEISUNGSSYNTAX

IQB0309 (&00) SDF: DIALOG ERROR CORRECTION NOT POSSIBLE  
IQB0309 (&00) SDF: FEHLER-KORREKTUR IM DIALOG NICHT MOEGLICH

IQB0310 (&00) SDF: LOGGING AREA TOO SMALL  
IQB0310 (&00) SDF: LOGGING BEREICH ZU KLEIN

IQB0311 (&00) SDF: UNEXPECTED END-STATEMENT FOUND  
IQB0311 (&00) SDF: UNERWARTETE END-ANWEISUNG GEFUNDEN

IQB0312 (&00) SDF: DIALOG ERROR CORRECTION REFUSED  
IQB0312 (&00) SDF: DIALOG FEHLER KORREKTUR ZURUECKGEWIESEN

IQB0313 (&00) SDF: NOT AVAILABLE FOR EXECUTION  
IQB0313 (&00) SDF: NICHT ZUR AUSFUEHRUNG VERFUEGBAR

IQB0314 (&00) SDF: PROGRAM NOT KNOWN IN SYNTAX FILE  
IQB0314 (&00) SDF: PROGRAMM NICHT IN SYNTAX-DATEI

IQB0315 (&00) SDF: NOT LOADED IN MEMORY  
IQB0315 (&00) SDF: NICHT IM SPEICHER GELADEN

IQB0316 (&00) SDF: STATEMENT NOT KNOWN OR WRONG SYNTAX FILE IN USE  
IQB0316 (&00) SDF: ANWEISUNG NICHT BEKANNT ODER FALSCHER SYNTAX-DATEI

IQB0317 (&00) TOO MANY PRECOMPILE STATEMENTS ((&01)) READ FROM SYSSTMT  
IQB0317 (&00) ZU VIELE PRECOMPILE ANWEISUNGEN ((&01)) VON SYSSTMT GELESEN

IQB0318 (&00) \*SUBSTITUTE IDENTIFIER-TOKEN <(&01)> DUPLICATE  
IQB0318 (&00) \*SUBSTITUTE BEZEICHNER <(&01)> ZWEIDEUTIG

IQB0319 (&00) \*SUBSTITUTE REPLACEMENT-TOKEN <(&01)> FOR NULL IDENTIFIER-TOKEN  
IQB0319 (&00) \*SUBSTITUTE ERSETZUNG <(&01)> FUER LEEREN BEZEICHNER

IQB0320 (&00) ESQL-DIALECT=ISO AND QUOTATION-CHARACTER=HOST-STD CONFLICT  
IQB0320 (&00) KONFLIKT ESQL-DIALECT=ISO UND QUOTATION-CHARACTER=HOST-STD

IQB0321 (&00) CATALOG AND SCHEMA OPTIONS MUST BE EXPLICITLY ASSIGNED  
IQB0321 (&00) CATALOG UND SCHEMA OPTIONEN MUESSEN EXPLIZIT ANGEGEBEN WERDEN

IQB0330 (&00) SQLOPT (V1.0B) STATEMENTS FOUND IN ESQL-HOST-PROGRAM  
IQB0330 (&00) SQLOPT (V1.0B) ANWEISUNGEN IN ESQL-HOST-PROGRAMM GEFUNDEN

IQB0350 (&00) PREMATURE END OF ESQL-HOST-PROGRAM INPUT FILE  
 IQB0350 (&00) VORZEITIGES ENDE DER EINGABEDATEI FUER DAS ESQL-HOST-PROGRAMM  
 IQB0351 (&00) CONFLICTING OR UNORDERED OPTIONS IN ESQL-HOST-PROGRAM  
 IQB0351 (&00) WIDERSPRUECHLICHE ODER UNGEORDNETE OPTIONEN IN ESQL-HOST-PROG.  
 IQB0352 (&00) //PRECOMPILE ... OPTIONS EXCEED SDF TRANSLATION-AREA SIZE  
 IQB0352 (&00) UEBERLAUF SDF-UEBERS.-BEREICH DURCH //PRECOMPILE ... OPTIONEN  
 IQB0357 (&00) CONTINUATION LINE EXPECTED IN ESQL-HOST-PROGRAM LINE (&01)  
 IQB0357 (&00) FORTSETZUNGSZEILE ERWARTET: ESQL-HOST-PROGRAMM ZEILE (&01)  
 IQB0358 (&00) //PRECOMPILE NOT TERMINATED BY //END IN ESQL-HOST-PROGRAM  
 IQB0358 (&00) //PRECOMPILE IM ESQL-HOST-PROGRAMM NICHT MIT //END ABGESCHLOSSEN  
 IQB0359 (&00) INCORRECT //PRECOMPILE ... OPTIONS IN ESQL-HOST-PROGRAM  
 IQB0359 (&00) FALSCHES //PRECOMPILE ... OPTIONEN IM ESQL-HOST-PROGRAMM  
 IQB0360 (&00) COBRUN/COMOPT NOT TERMINATED BY END IN ESQL-HOST-PROGRAM  
 IQB0360 (&00) COBRUN/COMOPT IM ESQL-HOST-PROGRAMM NICHT MIT END ABGESCHLOSSEN  
 IQB0361 (&00) //COMPILE ... NOT TERMINATED BY //END IN ESQL-HOST-PROGRAM  
 IQB0361 (&00) //COMPILE ... IM ESQL-HOST-PROGRAMM NICHT MIT //END ABGESCHLOSSEN  
 IQB0362 (&00) UNEXPECTED SQLOPT OPTION IN ESQL-HOST-PROGRAM LINE (&02): (&01)  
 IQB0362 (&00) UNERWARTETE SQLOPT-OPTION IM ESQL-HOST-PROG. ZEILE (&02): (&01)  
 IQB0370 (&00) REPLACE-BY-FILE SYNONYM-INPUT-FILE IS EMPTY  
 IQB0370 (&00) REPLACE-BY-FILE SYNONYM-EINGABEDATEI IST LEER  
 IQB0371 (&00) //PRECOMPILE ... STATEMENT NOT FOUND IN REPLACE-BY-FILE INPUT  
 IQB0371 (&00) //PRECOMPILE ... ANWEISG. IN REPLACE-BY-FILE EING. NICHT GEFUNDEN  
 IQB0372 (&00) //PRECOMPILE ... IN REPLACE-BY-FILE INPUT NOT TERMINATED BY //END  
 IQB0372 (&00) //PRECOMPILE ... IN REPLACE-BY-FILE EING. NICHT MIT //END ABGESCH  
 IQB0373 (&00) INCONSISTENCIES IN SDF-SYNTAX-FILE (REPLACE-BY-FILE)  
 IQB0373 (&00) SDF-SYNTAX-DATEI INKONSISTENT (REPLACE-BY-FILE)  
 IQB0374 (&00) UNRECOGNIZABLE DATA IN INPUT (REPLACE-BY-FILE)  
 IQB0374 (&00) UNBEKANNTE EINGABEDATEN (REPLACE-BY-FILE)  
 IQB0375 (&00) CONTINUATION LINE EXPECTED IN REPLACE-BY-FILE LINE (&01)  
 IQB0375 (&00) FORTSETZUNGSZEILE ERWARTET: REPLACE-BY-FILE, ZEILE (&01)  
 IQB0376 (&00) INCORRECT //PRECOMPILE ... OPTIONS IN INPUT (REPLACE-BY-FILE)  
 IQB0376 (&00) FALSCHES //PRECOMPILE ... OPTIONEN (REPLACE-BY-FILE)  
 IQB0501 (&00) WRITE FAILURE ON OBJECT MODULE LIBRARY (INSUFFICIENT MEMORY)  
 IQB0501 (&00) SCHREIBFEHLER OBJEKT-MODUL-BIBLIOTHEK (SPEICHERMANGEL)  
 IQB0502 (&00) WRITE FAILURE ON OBJECT MODULE LIBRARY (REASON (&01))  
 IQB0502 (&00) SCHREIBFEHLER OBJEKT-MODUL-BIBLIOTHEK (GRUND (&01))  
 IQB0601 (&00) SESMOD/SES LINK NOT FOUND DURING DYNAMIC LOAD  
 IQB0601 (&00) SESMOD/SES LINK BEIM DYNAMISCHEN LADEN NICHT GEFUNDEN

IQB0701 (&00) ERROR LIMIT REACHED BEFORE ALL ERRORS COULD BE FLAGGED  
 IQB0701 (&00) OBERGRENZE FEHLERANZAHL UEBERSCHRITTEN  
  
 IQB0900 (&00) INTERNAL ERROR: (&01) WHILST OUTPUTTING OBJECT MODULE  
 IQB0900 (&00) INTERNER FEHLER: (&01) WAEHREND OBJEKT-MODUL-AUSGABE  
  
 IQB0901 (&00) (&01)  
 IQB0901 (&00) (&01)  
  
 IQB0903 (&00) INTERNAL ERROR: <(&01)>. REASON (&02)  
 IQB0903 (&00) INTERNER FEHLER: <(&01)>. GRUND (&02)  
  
 IQB0905 (&00) INTERNAL ERROR: <(&01)> (&02)  
 IQB0905 (&00) INTERNER FEHLER: <(&01)> (&02)  
  
 IQB0907 (&00) INTERNAL ERROR: <(&01)> <(&02)>  
 IQB0907 (&00) INTERNER FEHLER: <(&01)> <(&02)>  
  
 IQB0909 (&00) INTERNAL ERROR: (&01)  
 IQB0909 (&00) INTERNER FEHLER: (&01)  
  
 IQB0910 (&00)(&01)  
 IQB0910 (&00)(&01)  
  
 IQB0921 (&00) INTERNAL ERROR: (&01) (&02)  
 IQB0921 (&00) INTERNER FEHLER: (&01) (&02)  
  
 IQB0931 (&00) INTERNAL ERROR: DC, (&01) TOO LONG OPTION-IDENTIFIER  
 IQB0931 (&00) INTERNER FEHLER: DC, (&01) OPTIONEN-BEZEICHNER ZU LANG  
  
 IQB0932 (&00) INTERNAL ERROR: DC, (&01) SQL-SYSTEM-FAILURE RC=(&02)/EC=(&03)  
 IQB0932 (&00) INTERNER FEHLER: DC,(&01): FEHLER IM SQL-SYSTEM RC=(&02)/EC=(&03)  
  
 IQB0933 (&00) INTERNAL ERROR: DC, (&01): UNKNOWN COMBINATION RC=(&02)/EC=(&03)  
 IQB0933 (&00) INTERNER FEHLER: DC,(&01): UNBEK. KOMBINATION RC=(&02)/EC=(&03)  
  
 IQB0934 (&00) INTERNAL ERROR: DC, (&01): UNDEF. REACTION FOR RC=(&02)/EC=(&03)  
 IQB0934 (&00) INTERNER FEHLER: DC,(&01): UNDEF. REAKTION FUER RC=(&02)/EC=(&03)  
  
 IQB0935 (&00) INTERNAL ERROR: DC, AT TO ICSQLE CONVERSION FOR (&01)  
 IQB0935 (&00) INTERNER FEHLER: DC, AT NACH ICSQLE UMSETZUNG FUER (&01)  
  
 IQB0936 (&00) INTERNAL ERROR: DC, ICSQLE TO AT CONVERSION  
 IQB0936 (&00) INTERNER FEHLER: DC, ICSQLE NACH AT UMSETZUNG  
  
 IQB0937 (&00) INTERNAL ERROR: DC, PARAMETER-NAME ICSQLE TO AT CONVERSION  
 IQB0937 (&00) INTERNER FEHLER: DC, PARAMETER-NAME ICSQLE NACH AT UMSETZUNG  
  
 IQB0938 (&00) INTERNAL ERROR: DC, DATATYPE ICSQLE TO AT CONVERSION RESULT (&01)  
 IQB0938 (&00) INTERNER FEHLER: DC, DATATYPE ICSQLE NACH AT; ERGEBNIS (&01)  
  
 IQB0939 (&00) INTERNAL ERROR: DC, TOO MANY SUBSEQUENT CALLS TO (&01)  
 IQB0939 (&00) INTERNER FEHLER: DC, ZU VIELE AUFEINANDERFOLG. AUFRUFE VON (&01)  
  
 IQB0940 (&00) INTERNAL ERROR: DC, SQLROW OUT OF RANGE <(&01)>  
 IQB0940 (&00) INTERNER FEHLER: DC, BEREICHSUEBERSCHREITUNG SQLROW: <(&01)>

IQB0941 (&00) INTERNAL ERROR: DC, USAGE ICSQLE TO AT CONVERSION RESULT (&01)  
 IQB0941 (&00) INTERNER FEHLER: DC, USAGE ICSQLE NACH AT ; ERGEBNIS (&01)  
 IQB0942 (&00) INTERNAL ERROR: DC, CATALOG-NAME ICSQLE TO NT CONVERSION  
 IQB0942 (&00) INTERNER FEHLER: DC, CATALOG-NAME ICSQLE NACH AT UMSETZUNG  
 IQB0943 (&00) INTERNAL ERROR: DC, VARIABLE BUFFER SIZE INCONSISTENCY  
 IQB0943 (&00) INTERNER FEHLER: DC, INKONSISTENZ IN VARIABLEM PUFFER  
 IQB0945 (&00) INTERNAL ERROR: HG, LABEL-NAME TOO LONG <(&01)>  
 IQB0945 (&00) INTERNER FEHLER: HG, MARKEN-NAME ZU LANG <(&01)>  
 IQB0946 (&00) INTERNAL ERROR: HG, NODE <KIND ZERO> DETECTED  
 IQB0946 (&00) INTERNER FEHLER: HG, KNOTEN VOM TYP <ZERO> ENTDECKT  
 IQB0947 (&00) INTERNAL ERROR: HG, IDMARK TOO LONG <(&01)>  
 IQB0947 (&00) INTERNER FEHLER: HG, IDMARK ZU LANG <(&01)>  
 IQB0948 (&00) INTERNAL ERROR: HG, ENTRY-NAME TOO LONG <(&01)>  
 IQB0948 (&00) INTERNER FEHLER: HG, ENTRY NAME ZU LANG <(&01)>  
 IQB0949 (&00) INTERNAL ERROR: HG, FORGOTTEN MESSAGE FOR LINE <(&01)> FOUND  
 IQB0949 (&00) INTERNER FEHLER: HG, VERGESSENE MELDUNG FUER ZEILE <(&01)>  
 IQB0950 (&00) INTERNAL ERROR: HG, INCLUDE STACK <(&01)>  
 IQB0950 (&00) INTERNER FEHLER: HG, INCLUDE STACK <(&01)>  
 IQB0951 (&00) INTERNAL ERROR: HG, UNKNOWN EXCEPTION-CONDITION <(&01)> FROM AT  
 IQB0951 (&00) INTERNER FEHLER: HG, UNBEKANNTE EXCEPTION-BEDINGUNG <(&01)>  
 IQB0952 (&00) INTERNAL ERROR: HG, WRONG STATE <(&01)> IN FUNCTION <(&02)>  
 IQB0952 (&00) INTERNER FEHLER: HG, FALSCHER ZUSTAND <(&01)> IN FUNKTION <(&02)>  
 IQB0953 (&00) INTERNAL ERROR: HG, MORE THAN ONE COLLISION RANGE FOUND  
 IQB0953 (&00) INTERNER FEHLER: HG, MEHR ALS EINE COLLISION RANGE GEFUNDEN  
 IQB0955 (&00) INTERNAL ERROR: LC, OPERATION (&01): LINE-NUMBER OUT OF SEQUENCE  
 IQB0955 (&00) INTERNER FEHLER: LC, OPERATION (&01): FEHLER IN ZEILENNUMMER  
 IQB0956 (&00) INTERNAL ERROR: LC, OPERATION (&01): RANGE NOT SET  
 IQB0956 (&00) INTERNER FEHLER: LC, OPERATION (&01): RANGE NICHT GESETZT  
 IQB0977 (&00) INTERNAL ERROR: AT, (&01)  
 IQB0977 (&00) INTERNER FEHLER: AT, (&01)  
 IQB1001 (&00) INVALID EXCEPTION CONDITION  
 IQB1001 (&00) FALSCHER AUSNAHME-BEDINGUNG  
 IQB1002 (&00) <END DECLARE SECTION> EXPECTED  
 IQB1002 (&00) <END DECLARE SECTION> ERWARTET  
 IQB1003 (&00) INVALID TOKEN; CONTINUE, GO TO OR GOTO EXPECTED  
 IQB1003 (&00) FALSCHES TOKEN; CONTINUE, GO TO ODER GOTO ERWARTET  
 IQB1004 (&00) INVALID TOKEN; TO EXPECTED  
 IQB1004 (&00) FALSCHES TOKEN; TO ERWARTET

IQB1005 (&00) INVALID TOKEN; FOUND EXPECTED  
IQB1005 (&00) FALSCHES TOKEN; FOUND ERWARTET

IQB1006 (&00) RIGHT-HAND DELIMITER <QUOTE> EXPECTED  
IQB1006 (&00) RECHTE BEGRENZUNG <QUOTE> ERWARTET

IQB1008 (&00) <BEGIN DECLARE SECTION> MISSING  
IQB1008 (&00) <BEGIN DECLARE SECTION> FEHLT

IQB1010 (&00) TOO LONG COBOL DATA-NAME  
IQB1010 (&00) COBOL DATEN-NAME ZU LANG

IQB1011 (&00) LABEL NAME EXPECTED  
IQB1011 (&00) MARKEN-NAME ERWARTET

IQB1012 (&00) INVALID SCALE; (3) EXPECTED  
IQB1012 (&00) FALSCHER SKALIERUNG; (3) ERWARTET

IQB1013 (&00) <INCLUDE> NOT ISO-CONFORM  
IQB1013 (&00) <INCLUDE> NICHT ISO-KONFORM

IQB1014 (&00) INVALID OR MISSING <INCLUDE> SPECIFICATION  
IQB1014 (&00) FALSCHER ODER FEHLENDE <INCLUDE> SPEZIFIKATION

IQB1015 (&00) INVALID ESQL STATEMENT  
IQB1015 (&00) FALSCHER ESQL-ANWEISUNG

IQB1016 (&00) INVALID OR MISSING COBOL DATA-NAME  
IQB1016 (&00) FALSCHER ODER FEHLENDER COBOL DATEN-NAME

IQB1017 (&00) TOO LONG PICTURE STRING  
IQB1017 (&00) PICTURE-ZEICHENFOLGE ZU LANG

IQB1018 (&00) INVALID OR MISSING DATA-ITEM PICTURE SIZE SPECIFICATION  
IQB1018 (&00) FALSCHER ODER FEHLENDE SPEZIFIKATION DER PICTURE-GROESSE

IQB1019 (&00) PICTURE STRING NOT ISO-CONFORM  
IQB1019 (&00) PICTURE-ZEICHENFOLGE NICHT ISO-KONFORM

IQB1020 (&00) INVALID OR MISSING COBOL PROCEDURE-NAME  
IQB1020 (&00) FALSCHER ODER FEHLENDER COBOL PROZEDUR-NAME

IQB1021 (&00) FILE TO BE INCLUDED NOT FOUND  
IQB1021 (&00) ZU INCLUDIERENDE DATEI NICHT GEFUNDEN

IQB1022 (&00) TOO MANY NESTED FILES FOR INCLUSION  
IQB1022 (&00) ZU VIELE GESCHACHELTE DATEIEN FUER <INCLUDE>

IQB1023 (&00) REPLACEMENT OBJECT LENGTH EXCEEDS LIMIT  
IQB1023 (&00) ERSETZUNG UEBERSCHREITET MAXIMALLAENGE

IQB1024 (&00) <INCLUDE> WITHIN REPLACEMENT OBJECT NOT ALLOWED  
IQB1024 (&00) <INCLUDE> IN ERSETZUNG NICHT ERLAUBT

IQB1025 (&00) <WHENEVER> WITHIN REPLACEMENT OBJECT NOT ALLOWED  
IQB1025 (&00) <WHENEVER> IN ERSETZUNG NICHT ERLAUBT

IQB1026 (&00) INVALID REPLACEMENT OBJECT  
 IQB1026 (&00) FALSCHER ERSETZUNG  
 IQB1027 (&00) DELIMITED IDENTIFIER NOT ENDED BY <QUOTE> IN REPLACEMENT OBJECT  
 IQB1027 (&00) <DELIMITED IDENTIFIER> IN ERSETZUNG NICHT MIT <QUOTE> ABGESCHLOS.  
 IQB1028 (&00) IDENTIFIER EXPECTED AFTER < . > IN REPLACEMENT OBJECT  
 IQB1028 (&00) BEZEICHNER NACH < . > IN ERSETZUNG ERWARTET  
 IQB1029 (&00) LEVEL NUMBER NOT SPECIFIED AS 1 OR 2 DIGIT(S)  
 IQB1029 (&00) <LEVEL NUMBER> NICHT MIT 1 ODER 2 ZIFFERN SPEZIFIZIERT  
 IQB1030 (&00) PARSER STACK EXAUSTED  
 IQB1030 (&00) PARSER STACK UEBERLAUF  
 IQB1031 (&00) INVALID INDEX OR RANGE  
 IQB1031 (&00) FALSCHER INDEX ODER BEREICH  
 IQB1032 (&00) UNSIGNED INTEGER EXPECTED AFTER < ( >  
 IQB1032 (&00) <UNSIGNED INTEGER> NACH < ( > ERWARTET  
 IQB1033 (&00) INVALID OR MISSING COBOL PROCEDURE-NAME  
 IQB1033 (&00) FALSCHER ODER FEHLENDER COBOL PROZEDUR-NAME  
 IQB1034 (&00) END-EXEC EXPECTED  
 IQB1034 (&00) END-EXEC ERWARTET  
 IQB1035 (&00) INVALID COBOL DATA-ITEM DEFINITION  
 IQB1035 (&00) FALSCHER COBOL DATA-ITEM DEFINITION  
 IQB1036 (&00) QUALIFICATION OF COBOL IDENTIFIER NOT ISO-CONFORM  
 IQB1036 (&00) QUALIFIZIERUNG DES COBOL BEZEICHNERS NICHT ISO-KONFORM  
 IQB1037 (&00) INDEX NOT ISO-CONFORM  
 IQB1037 (&00) INDEX NICHT ISO-KONFORM  
 IQB1038 (&00) INVALID <INCLUDE>  
 IQB1038 (&00) FALSCHES <INCLUDE>  
 IQB1039 (&00) LEVEL NUMBER OUT OF RANGE  
 IQB1039 (&00) <LEVEL NUMBER> AUSSERHALB DES ZULAESSIGEN BEREICHS  
 IQB1040 (&00) INVALID PICTURE STRING  
 IQB1040 (&00) FALSCHER PICTURE-ZEICHENFOLGE  
 IQB1041 (&00) PERIOD < . > EXPECTED  
 IQB1041 (&00) PUNKT < . > ERWARTET  
 IQB1042 (&00) INVALID TOKEN; LEADING OR TRAILING EXPECTED  
 IQB1042 (&00) FALSCHES TOKEN; LEADING ODER TRAILING ERWARTET  
 IQB1043 (&00) INVALID OR MISSING USAGE CLAUSE  
 IQB1043 (&00) FALSCHER ODER FEHLENDE USAGE-KLAUSEL  
 IQB1044 (&00) INVALID OR MISSING OCCURS CLAUSE  
 IQB1044 (&00) FALSCHER ODER FEHLENDE OCCURS-KLAUSEL

IQB1045 (&00) INVALID VALUE CLAUSE  
IQB1045 (&00) FALSCHER VALUE-KLAUSEL

IQB1046 (&00) INVALID TIMESTAMP SPECIFICATION  
IQB1046 (&00) FALSCHER TIMESTAMP-SPEZIFIKATION

IQB1047 (&00) INVALID TIME SPECIFICATION  
IQB1047 (&00) FALSCHER TIME-SPEZIFIKATION

IQB1048 (&00) LEVEL NUMBER EXPECTED  
IQB1048 (&00) STUFEN-NUMMER ERWARTET

IQB1049 (&00) INVALID OR MISSING COBOL DATA-NAME  
IQB1049 (&00) FALSCHER ODER FEHLENDER COBOL DATEN-NAME

IQB1050 (&00) INVALID OR MISSING PICTURE CLAUSE  
IQB1050 (&00) FALSCHER ODER FEHLENDE PICTURE-KLAUSEL

IQB1051 (&00) INVALID OR MISSING SIGN CLAUSE  
IQB1051 (&00) FALSCHER ODER FEHLENDE SIGN-KLAUSEL

IQB1052 (&00) INVALID OR MISSING LEADING CLAUSE  
IQB1052 (&00) FALSCHER ODER FEHLENDE LEADING-KLAUSEL

IQB1053 (&00) INVALID OR MISSING SEPARATE CLAUSE  
IQB1053 (&00) FALSCHER ODER FEHLENDE SEPARATE-KLAUSEL

IQB2001 (&00) SIGN CLAUSE NOT ALLOWED  
IQB2001 (&00) SIGN-KLAUSEL NICHT ERLAUBT

IQB2002 (&00) ONLY USAGE DISPLAY ALLOWED  
IQB2002 (&00) NUR USAGE DISPLAY ERLAUBT

IQB2003 (&00) SIGN CLAUSE ONLY ALLOWED WITH USAGE DISPLAY  
IQB2003 (&00) SIGN-KLAUSEL NUR ERLAUBT MIT USAGE DISPLAY

IQB2004 (&00) NO PICTURE CLAUSE ALLOWED WITH COMP-1 OR COMP-2  
IQB2004 (&00) KEINE PICTURE-KLAUSEL ERLAUBT MIT COMP-1 ODER COMP-2

IQB2005 (&00) USAGE BINARY OR COMP IN SQL ONLY ALLOWED FOR INTEGERS  
IQB2005 (&00) USAGE BINARY ODER COMP IN SQL NUR FUER GANZE ZAHLEN ERLAUBT

IQB2006 (&00) TOO MANY DIGIT POSITIONS FOR USAGE BINARY OR COMP  
IQB2006 (&00) ZU VIELE ZIFFERN-POSITIONEN FUER USAGE BINARY ODER COMP

IQB2007 (&00) INCORRECT LEVEL NUMBER  
IQB2007 (&00) FALSCHER STUFEN-NUMMER

IQB2008 (&00) INCORRECT LEVEL NUMBER : LEVEL NUMBER OUT OF RANGE  
IQB2008 (&00) FALSCHER STUFEN-NUMMER : BEREICHSUEBERSCHREITUNG

IQB2009 (&00) DUPLICATE PICTURE CLAUSE ILLEGAL  
IQB2009 (&00) MEHRFACHE PICTURE-KLAUSEL NICHT ERLAUBT

IQB2010 (&00) DUPLICATE SIGN CLAUSE ILLEGAL  
IQB2010 (&00) MEHRFACHE SIGN-KLAUSEL NICHT ERLAUBT

IQB2011 (&00) DUPLICATE SYNCHRONIZED CLAUSE ILLEGAL  
 IQB2011 (&00) MEHRFACHE SYNCHRONIZED-KLAUSEL NICHT ERLAUBT  
 IQB2012 (&00) DUPLICATE USAGE CLAUSE ILLEGAL  
 IQB2012 (&00) MEHRFACHE USAGE-KLAUSEL NICHT ERLAUBT  
 IQB2013 (&00) DUPLICATE EXTERNAL CLAUSE ILLEGAL  
 IQB2013 (&00) MEHRFACHE EXTERNAL-KLAUSEL NICHT ERLAUBT  
 IQB2014 (&00) DUPLICATE GLOBAL CLAUSE ILLEGAL  
 IQB2014 (&00) MEHRFACHE GLOBAL-KLAUSEL NICHT ERLAUBT  
 IQB2015 (&00) DUPLICATE OCCURS CLAUSE ILLEGAL  
 IQB2015 (&00) MEHRFACHE OCCURS-KLAUSEL NICHT ERLAUBT  
 IQB2016 (&00) DUPLICATE VALUE CLAUSE ILLEGAL  
 IQB2016 (&00) MEHRFACHE VALUE-KLAUSEL NICHT ERLAUBT  
 IQB2017 (&00) ILLEGAL NESTING OF OCCURS  
 IQB2017 (&00) FALSCHER VERSCHACHELUNG VON OCCURS  
 IQB2018 (&00) OCCURS CLAUSE NOT ALLOWED FOR THIS LEVEL NUMBER  
 IQB2018 (&00) OCCURS-KLAUSEL NICHT ERLAUBT FUER DIESE STUFENNUMMER  
 IQB2019 (&00) ILLEGAL ITEM SUBORDINATE TO VARCHAR  
 IQB2019 (&00) VARCHAR UNTERGEORDNETES DATENELEMENT/-GRUPPE NICHT ERLAUBT  
 IQB2020 (&00) CLAUSE ILLEGAL WITH VARCHAR  
 IQB2020 (&00) KLAUSEL NICHT ERLAUBT MIT VARCHAR  
 IQB2021 (&00) ITEM SUBORDINATE TO DATETIME ILLEGAL  
 IQB2021 (&00) DATE/TIME UNTERGEORDNETES DATENELEMENT/-GRUPPE NICHT ERLAUBT  
 IQB2022 (&00) CLAUSE ILLEGAL WITH DATETIME  
 IQB2022 (&00) KLAUSEL NICHT ERLAUBT MIT DATE/TIME  
 IQB2024 (&00) TOO FEW SUBORDINATE COMPONENTS  
 IQB2024 (&00) ZU WENIGE UNTERGEORDNETE KOMPONENTEN  
 IQB2025 (&00) VALUE ILLEGAL IN OCCURS CLAUSE  
 IQB2025 (&00) UNERLAUBTER WERT IN OCCURS-KLAUSEL  
 IQB2026 (&00) TOO BIG OCCURS VALUE  
 IQB2026 (&00) OCCURS-WERT ZU GROSS  
 IQB2027 (&00) ILLEGAL REDEFINITION OF HOST VARIABLE  
 IQB2027 (&00) UNERLAUBTE REDEFINITION EINER BENUTZER-VARIABLEN  
 IQB2028 (&00) COBOL TYPE NOT ALLOWED  
 IQB2028 (&00) COBOL-TYP NICHT ERLAUBT  
 IQB2029 (&00) CLAUSE NOT ALLOWED FOR GROUP ITEMS  
 IQB2029 (&00) KLAUSEL NICHT ERLAUBT FUER DATENGRUPPEN  
 IQB2030 (&00) HOST VARIABLE UNDEFINED  
 IQB2030 (&00) BENUTZER-VARIABLE NICHT DEFINIERT

IQB2031 (&00) AMBIGUOUS QUALIFICATION  
IQB2031 (&00) MEHRDEUTIGE QUALIFIZIERUNG

IQB2032 (&00) ILLEGAL GROUP REFERENCE IN HOST IDENTIFIER  
IQB2032 (&00) UNERLAUBTE REFERENZ AUF EINE DATENGRUPPE IN BENUTZER-VARIABLE

IQB2033 (&00) SUBSCRIPT OUT OF RANGE  
IQB2033 (&00) BEREICHSUEBERSCHREITUNG IN SUBSKRIPT

IQB2034 (&00) SUBSCRIPT ON ITEM NOT SUBORDINATE TO OCCURS  
IQB2034 (&00) SUBSKRIBIERTE DATENELEMENT/-GRUPPE NICHT OCCURS UNTERGEORDNET

IQB2035 (&00) ILLEGAL SUBSCRIPT RANGE  
IQB2035 (&00) UNERLAUBTER BEREICH IN SUBSKRIPT

IQB2036 (&00) TYPE OF HOST VARIABLE NOT ALLOWED IN SQL STATEMENT  
IQB2036 (&00) BENUTZER-VARIABLEN-TYP NICHT ERLAUBT IN SQL-ANWEISUNG

IQB2037 (&00) TOO MANY QUALIFICATIONS IN HOST IDENTIFIER  
IQB2037 (&00) ZU VIELE QUALIFIZIERUNGEN IN BENUTZER-VARIABLE

IQB2038 (&00) INDEX OR INDEX RANGE MISSING  
IQB2038 (&00) FEHLENDER INDEX ODER INDEX-BEREICH

IQB2040 (&00) INCORRECT TYPE FOR SQLCODE  
IQB2040 (&00) FALSCHER TYP FUER SQLCODE

IQB2041 (&00) ILLEGAL DECLARATION OF SQLCODE AFTER SQL STATEMENT  
IQB2041 (&00) DEKLARATION VON SQLCODE NACH SQL-ANWEISUNG NICHT ERLAUBT

IQB2042 (&00) INCORRECT TYPE FOR SQLSTATE  
IQB2042 (&00) FALSCHER TYP FUER SQLSTATE

IQB2043 (&00) ILLEGAL DECLARATION OF SQLSTATE AFTER SQL STATEMENT  
IQB2043 (&00) DEKLARATION VON SQLSTATE NACH SQL-ANWEISUNG NICHT ERLAUBT

IQB2044 (&00) SQLCODE MUST NOT BE A GROUP ITEM  
IQB2044 (&00) SQLCODE DARF KEINE DATENGRUPPE SEIN

IQB2045 (&00) SQLCODE MUST NOT CONTAIN, NOR BE SUBORDINATE TO OCCURS  
IQB2045 (&00) SQLCODE DARF KEIN OCCURS ENTHALTEN ODER OCCURS UNTERGEORDNET SEIN

IQB2046 (&00) SQLSTATE MUST NOT BE A GROUP ITEM  
IQB2046 (&00) SQLSTATE DARF KEINE DATENGRUPPE SEIN

IQB2047 (&00) SQLSTATE MUST NOT CONTAIN, NOR BE SUBORDINATE TO OCCURS  
IQB2047 (&00) SQLSTATE DARF KEIN OCCURS ENTHALTEN ODER UNTERGEORDNET SEIN

IQB2048 (&00) SQLCODE ALREADY DEFINED  
IQB2048 (&00) SQLCODE BEREITS DEFINIERT

IQB2049 (&00) SQLSTATE ALREADY DEFINED  
IQB2049 (&00) SQLSTATE BEREITS DEFINIERT

IQB2050 (&00) DATA DESCRIPTION NOT CONFORMING WITH ISO SQL  
IQB2050 (&00) DATEN BESCHREIBUNG NICHT ISO-SQL-KONFORM

IQB2051 (&00) HOST VARIABLE NOT CONFORMING WITH ISO SQL  
 IQB2051 (&00) BENUTZERVARIABLE NICHT ISO-SQL-KONFORM  
 IQB2052 (&00) ILLEGAL ITEM SUBORDINATE TO NVARCHAR  
 IQB2052 (&00) ILLEGALES FELD UNTER NVARCHAR  
 IQB2053 (&00) CLAUSE ILLEGAL WITH NVARCHAR  
 IQB2053 (&00) DIESE KLAUSEL IST MIT NVARCHR ILLEGAL  
 IQB2054 (&00) ONLY USAGE NATIONAL ALLOWED  
 IQB2054 (&00) NUR USAGE NATIONAL ERLAUBT  
 IQB3000 (&00) (&01)  
 IQB3000 (&00) (&01)  
 IQB3001 (&00) EMBEDDED SQL STATEMENT TOO LONG  
 IQB3001 (&00) ESQL-ANWEISUNG ZU LANG  
 IQB3002 (&00) EMBEDDED SQL STATEMENT CONTAINS TOO MANY HOST VARIABLES  
 IQB3002 (&00) ESQL-ANWEISUNG ENTHAELT ZU VIELE BENUTZER-VARIABLEN  
 IQB9001 (&00)BEGIN (&01)  
 IQB9001 (&00)BEGIN (&01)  
 IQB9002 (&00)COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1994.  
 IQB9002 (&00)COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1994.  
 IQB9003 (&00)ALL RIGHTS RESERVED.  
 IQB9003 (&00)ALLE RECHTE VORBEHALTEN.  
 IQB9004 (&00)MESSAGE STATISTICS: (&01) NOTES, (&02) WARNINGS, (&03) ERRORS  
 IQB9004 (&00)MELDUNGS-STATISTIK: (&01) NOTES, (&02) WARNINGS, (&03) ERRORS  
 IQB9006 (&00)MODULE NOT GENERATED  
 IQB9006 (&00)MODUL NICHT ERZEUGT  
 IQB9007 (&00)MODULE GENERATED  
 IQB9007 (&00)MODUL ERZEUGT  
 IQB9008 (&00)END (&01) -- TIME USED = (&02) SECS.  
 IQB9008 (&00)END (&01) -- ZEITVERBRAUCH = (&02) SECS.  
 SIS0001 COMP=SISMF,VER=100,DATE=930131,COMPNR=00000000  
 SIS0001 COMP=SISMF,VER=100,DATE=930131,COMPNR=00000000  
 SIS0003 No previous error occured  
 SIS0003 Es ist noch kein Fehler aufgetreten  
 SIS0004 Not enough or too many inserts passed  
 SIS0004 Es wurden zu wenige oder zu viele Einschuebe uebergeben

**Bedeutung**

Fehlende Einschuebe werden durch die leere Zeichenfolge ersetzt.

Ueberfluessige Einschuebe werden ignoriert.

SIS0005 PROSYS messages are not available  
SIS0005 PROSYS Meldungen sind nicht verfuegbar

SIS0006 Version of message class (&00) is wrong  
SIS0006 Version der Meldungsklasse (&00) ist fehlerhaft

SIS0007 Version of set (&00) in the specified catalog is wrong  
SIS0007 Version der Meldungsmenge (&00) im spezifizierten Katalog ist fehlerhaft

SIS0008 Message (&00)(&01) is not defined  
SIS0008 Meldung (&00)(&01) ist nicht definiert

SIS0009 In the specified catalog message (&00),(&01) is not defined  
SIS0009 Im spezifizierten Katalog ist die Meldung (&00),(&01) nicht definiert

SIS0010 Message catalog (&00) can not be opened  
SIS0010 Fehler beim Oeffnen des Meldungskataloges (&00)

SIS0011 Parameter (&00) is wrong  
SIS0011 Parameter (&00) ist fehlerhaft

SIS0012 System specific error occured  
SIS0012 Es ist ein systemspezifischer Fehler aufgetreten

SIS0036 PROSOS internal Error  
SIS0036 PROSOS interner Fehler

SIS0100 Error during initialisation of SHS  
SIS0100 Fehler bei der Initialisierung von SHS

SIS0101 reserved  
SIS0101 reserviert

SIS0102 Insufficient memory  
SIS0102 Speichermangel aufgetreten

SIS0103 Invalid size parameter -- (&00)  
SIS0103 Ungueltige Angabe der Speichergroesse -- (&00)

SIS0104 Size (&00) too big  
SIS0104 Angabe der Speichergroesse (&00) zu gross

SIS0105 Option requires an argument -- (&00)  
SIS0105 Option erwartet ein Argument -- (&00)

SIS0106 Illegal option -- (&00)  
SIS0106 Ungueltige Option -- (&00)

SIS0107 Handle does not point to string container  
SIS0107 Handle zeigt nicht auf einen String-Container

SIS0108 Invalid close mode  
SIS0108 Unzulaessiger Close-Modus

SIS0109 String container cannot be opened  
SIS0109 String-Container kann nicht geoeffnet werden

SIS0110 Invalid access of a string container  
SIS0110 Unzulaessiger Zugriff auf einen String-Container

SIS0111 Write access of a string container not allowed  
SIS0111 Schreibender Zugriff auf den String-Container ist nicht erlaubt

SIS0112 System specific error ocured  
SIS0112 Es ist ein systemspezifischer Fehler aufgetreten

SIS0200 No more space available  
SIS0200 Kein ausreichender Speicherplatz vorhanden

SIS0201 Error during object generation: (&00)  
SIS0201 Fehler bei der Objektgenerierung: (&00)

SIS0202 File (&00) cannot be opened  
SIS0202 Datei (&00) kann nicht geoeffnet werden

SIS0601 End of file detected  
SIS0601 Dateiende ist erreicht

SIS0602 Specified record does not exist  
SIS0602 Spezifizierter Satz existiert nicht

SIS0603 Specified record exists  
SIS0603 Spezifizierter Satz existiert bereits

SIS0604 Begin of file detected  
SIS0604 Dateianfang ist erreicht

SIS0605 Specified link does not exist  
SIS0605 Spezifizierter Link existiert nicht

SIS0606 Length of file name exceeds P\_MAXFILENAME  
SIS0606 Dateiname ist laenger als P\_MAXFILENAME

SIS0607 Length of path string exceeds P\_MAXPATHSTRG  
SIS0607 Pfad ist laenger als P\_MAXPATHSTRG

SIS0608 Length of path name exceeds P\_MAXPATHNAME  
SIS0608 Pfadname ist laenger als P\_MAXPATHNAME

SIS0609 Length of link name exceeds P\_MAXLINKNAME  
SIS0609 Linkname ist laenger als P\_MAXLINKNAME

SIS0610 No more space available  
SIS0610 Kein ausreichender Speicherplatz verfuegbar

SIS0611 Number of path elements exceeds P\_MAXHIERARCHY  
SIS0611 Anzahl der Pfadelemente uebersteigt P\_MAXHIERARCHY

SIS0612 Function not supported  
SIS0612 Funktion wird nicht unterstuetzt

SIS0613 Missing file name or syntax error in file name  
SIS0613 Dateiname ist fehlerhaft oder leer

SIS0614 Number of secondary keys exceeds P\_MAXKEYS  
SIS0614 Anzahl der Sekundaerschluessel uebersteigt P\_MAXKEYS

SIS0615 Number of open files exceeds system specific boundary  
SIS0615 Anzahl offener Dateien uebersteigt systemspezifische Grenze

SIS0616 Specified file does not exist  
SIS0616 Spezifizierte Datei existiert nicht

SIS0617 Write access not allowed  
SIS0617 Kein Schreibzugriff erlaubt

SIS0618 No file name found  
SIS0618 Kein Dateiname spezifiziert

SIS0619 File locked  
SIS0619 Datei ist gesperrt

SIS0620 Illegal combination of file attributes  
SIS0620 Unzulaessige Kombination von Dateiattributen

SIS0621 File handle is invalid  
SIS0621 File-Handle ist ungueltig

SIS0622 Current record smaller than MINSIZE  
SIS0622 Aktueller Satz ist kuerzer als MINSIZE

SIS0623 Current record bigger than MAXSIZE  
SIS0623 Aktueller Satz ist laenger als MAXSIZE

SIS0625 No read sequential performed before rewrite sequential  
SIS0625 Vor rewrite sequentiell wurde kein read sequentiell ausgefuehrt

SIS0626 Record format out of range or not allowed  
SIS0626 Spezifiziertes Satzformat ist unzulaessig

SIS0627 MINSIZE greater than MAXSIZE  
SIS0627 MINSIZE ist groesser als MAXSIZE

SIS0628 Organisation out of range or not allowed  
SIS0628 Spezifizierte Organisation ist unzulaessig

SIS0629 File exists but MUST\_NOT\_EXIST specified  
SIS0629 Nicht existent spezifizierte Datei existiert

SIS0630 Specified access function not allowed  
SIS0630 Spezifizierte Zugriffsfunktion ist nicht erlaubt

SIS0631 Key parameters out of range or not allowed  
SIS0631 Spezifizierter Schluessel ist unzulaessig

SIS0632 Duplicate key not allowed  
SIS0632 Mehrfachschluessel ist nicht erlaubt

SIS0633 Record currently locked  
SIS0633 Aktueller Satz ist zur Zeit gesperrt

SIS0634 Current key out of sequence  
SIS0634 Aktueller Schluessel in fehlerhafter Reihenfolge

SIS0635 Specified path undefined  
SIS0635 Spezifizierter Pfad ist undefiniert

SIS0637 End of line detected  
SIS0637 Zeilenende ist erreicht

SIS0638 Record truncated  
SIS0638 Satz wurde abgeschnitten

SIS0640 No more space available to extend the file  
SIS0640 Kein Speicherplatz zur Dateierweiterung verfuegbar

SIS0643 Open type out of range or not allowed  
SIS0643 Spezifizierter Oeffnungsmodus ist unzuellaessig

SIS0644 Length of link string exceeds P\_MAXLINKSTRG  
SIS0644 Laenge des Links uebersteigt P\_MAXFILESTRG

SIS0645 Invalid version identification specified  
SIS0645 Versionsidentifikation ist fehlerhaft

SIS0646 Existence out of range  
SIS0646 Spezifizierte Dateiexistenz ist unzuellaessig

SIS0647 Syntax error in file, link or path string  
SIS0647 Syntaxfehler im Dateinamen, Link oder Pfad

SIS0649 Close type out of range or not allowed  
SIS0649 Spezifizierter Modus beim Schliessen ist unzuellaessig

SIS0650 Access not permitted  
SIS0650 Dateizugriff ist nicht erlaubt

SIS0651 Parameter error  
SIS0651 Fehlerhafter Parameter angegeben

SIS0652 Invalid pointer to I/O area  
SIS0652 Zeiger in den Ein-/Ausgabebereich ist fehlerhaft

SIS0653 Invalid record length detected  
SIS0653 Satzlaenge ist fehlerhaft

SIS0654 Space limit on device reached  
SIS0654 Speichermangel auf Ausgabemedium aufgetreten

SIS0655 Print control out of range or not allowed  
SIS0655 Spezifizierte Vorschubsteuerung ist unzuellaessig

SIS0656 Code set out of range or not allowed  
SIS0656 Spezifizierter Code ist unzuellaessig

SIS0657 Combination of open type and existence not allowed  
SIS0657 Oeffnungsmodus und Dateiexistenz sind unzuellaessig kombiniert

SIS0658 I/O interrupted  
SIS0658 Ein-/Ausgabeunterbrechung ist aufgetreten

SIS0659 Length of keyword exceeds P\_MAXKEYWORD  
SIS0659 Laenge des Schluesselfwortes uebersteigt P\_MAXKEYWORD

SIS0660 Keyword ambiguous  
SIS0660 Schluesselfwort ist mehrdeutig

SIS0661 Number of exits exceeds P\_MAXEXITS  
SIS0661 Anzahl der Exits uebersteigt P\_MAXEXITS

SIS0662 New line character detected  
SIS0662 Zeilenvorschubsteuerzeichen erkannt

SIS0663 New page character detected  
SIS0663 Seitenvorschubsteuerzeichen erkannt

SIS0664 Not all pathes closed  
SIS0664 Einige Pfade sind nicht geschlossen

SIS0665 Next indexed record has same secondary key  
SIS0665 Naechster Satz hat den gleichen Sekundaerschluesel

SIS0666 Secondary key of written record exists already  
SIS0666 Sekundaerschluesel des geschriebenen Satzes existiert bereits

SIS0667 Current record number exceeds specified MAX\_REC\_NR  
SIS0667 Aktuelle Satznummer ist groesser als MAX\_REC\_NR

SIS0668 Path name exists already  
SIS0668 Pfad ist bereits definiert

SIS0669 Link name exists already  
SIS0669 Link ist bereits definiert

SIS0670 Positioning condition out of range  
SIS0670 Spezifizierter Wert fuer Positionierbedingung ist unzuellaessig

SIS0671 Unknown control character detected  
SIS0671 Unbekanntes Kontrollzeichen gefunden

SIS0672 No file name created  
SIS0672 Es konnte kein eindeutiger Dateiname erzeugt werden

SIS0673 Last partial record not completed  
SIS0673 Letzter Teilsatz wurde nicht abgeschlossen

SIS0674 Seek type out of range  
SIS0674 Spezifizierter Wert fuer Positionierung ist unzuellaessig

SIS0675 Record format not determinable  
SIS0675 Satzformat ist nicht bestimmbar

SIS0676 MAXSIZE not determinable  
SIS0676 MAXSIZE ist nicht bestimmbar

SIS0677 PROSOS-D internal error  
SIS0677 Interner PROSOS-D Fehler aufgetreten

SIS0678 Specified file is a container of files  
SIS0678 Spezifizierte Datei ist ein Container von Dateien

SIS0679 Specified file unreachable with given path  
SIS0679 Spezifizierte Datei ist unter angegebenem Pfad nicht erreichbar

SIS0680 Version not incrementable  
SIS0680 Versionsangabe kann nicht erhoeht werden

SIS0681 Reopen after implicit close failed  
SIS0681 Nochmaliges Oeffnen nach implizitem Schliessen wurde abgewiesen

SIS0682 Failure on initialisation of PROSOS-D  
SIS0682 Fehler bei der Initialisierung von PROSOS-D

SIS0683 Link indirections by extern variables exceed P\_MAXLINKNESTING  
SIS0683 Linkindirektionen uebersteigen P\_MAXLINKNESTING

SIS0901 Catalog handle is invalid  
SIS0901 Catalog-Handle ist ungueltig

SIS0902 Catalog or view (&00) not present  
SIS0902 Katalog oder Sicht eines Katalogs (&00) nicht vorhanden

SIS0903 End of catalog or view reached  
SIS0903 Ende des Katalogs oder Sicht erreicht

SIS0904 No more space available  
SIS0904 Kein ausreichender Speicherplatz verfuegbar

SIS0905 Function not supported  
SIS0905 Funktion wird nicht unterstuetzt

SIS0906 (&00) is file  
SIS0906 (&00) ist Datei



---

# 10 Anhang

Dieses Kapitel behandelt folgende Themen:

- Mischbetrieb von SQL- und CALL-DML-Schnittstelle
- Beispieldatenbank

## 10.1 Mischbetrieb von SQL- und CALL-DML-Schnittstelle

SESAM/SQL unterstützt die CALL-DML- und die SQL-Schnittstelle. In einer ESQL-COBOL-Anwendung können beide Schnittstellen zusammen verwendet werden (Mischbetrieb). Bis SESAM/SQL-Server V2.1 können innerhalb einer Transaktion die beiden Schnittstellen nicht gemischt werden. Ab SESAM/SQL-Server V2.2 dürfen SQL-Anweisungen in CALL-DML-Transaktionen auftreten

### Betriebsmittel

Im Mischbetrieb sprechen CALL-DML-Anweisungen nur Betriebsmittel an, die aufgrund von CALL-DML-Anweisungen im SESAM/SQL-DBH reserviert wurden. SQL-Anweisungen sprechen nur Betriebsmittel an, die aufgrund von SQL-Anweisungen reserviert wurden. Eine Anwendung, die beide Schnittstellen verwendet, hat sich ordnungsgemäß beim DBH abgemeldet, wenn die Betriebsmittel beider Schnittstellen freigegeben wurden.

### Statuscodes

Eine Anwendung gibt SQL-Statuscodes (SQLCODE oder SQLSTATE oder beide) zurück, wenn die letzte an den DBH übergebene Anweisung eine SQL-Anweisung war. War die letzte an den DBH übergebene Anweisung eine CALL-DML-Anweisung, gibt die Anwendung CALL-DML-Statuscodes zurück.

## 10.2 Beispieldatenbank

Die Beispiele in diesem Handbuch beziehen sich überwiegend auf die im folgenden beschriebene Beispieldatenbank. Zu jeder Tabelle werden die Definition der Tabelle und die Daten der Tabelle dargestellt. Die Beispieldatenbank enthält das Schema AUFTRAGSVER und das Schema TEILE.

### 10.2.1 Schema AUFTRAGSVER

Im Schema AUFTRAGSVER (Auftragsverwaltung) können Daten über Kunden, Kontaktpersonen, Aufträge und Leistungen verwaltet werden. Das Schema AUFTRAGSVER enthält die Tabellen KUNDE, KONTAKT, AUFTRAG, LEISTUNG und AUFSTAT.

### 10.2.1.1 Tabelle KUNDE

Die Tabelle KUNDE ist wie folgt definiert:

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

Die Tabelle KUNDE enthält folgende Daten:

knr	firma	strasse	plz	ort	land	ktelefon	info
100	Siemens AG	Otto-Hahn-Ring 6	81739	Muenchen	D	089/636-8	Elektro
101	Login GmbH	Rosenheimer Str. 34	81667	Muenchen	D	089/4488870	PC Netzwerke
102	JIKO GmbH	Posener Str. 12	)		D	0551/123874	Import-Export
103	Plenzer Trading	Paul-Heyse-Str. 12	80336	Muenchen	D	089/923764	Fruechtehandel
104	Jonas Fischladen	Hirschgartenstr. 12	12587	Berlin	D	016/5739921	Einzelhandel
105	Pudelshop Anke	Am Muehlentor 26	41179	Moenchengladbach	D	040/873562	Dienstleistung
106	Foreign Ltd.	26 West York St.		New York, NY	USA	001703/2386532	Handelsvertretung
107	Externa & Co KG	Berner Weg 78	3000	Bern 33	CH		Anwaltskanzlei

Tabelle 15: Daten der Tabelle KUNDE

### 10.2.1.2 Tabelle KONTAKT

Die Tabelle KONTAKT ist wie folgt definiert:

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

Die Tabelle KONTAKT enthält folgende Daten:

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

Tabelle 16: Daten der Tabelle KONTAKT

### 10.2.1.3 Tabelle AUFTRAG

Die Tabelle AUFTRAG ist wie folgt definiert:

```
CREATE TABLE    auftrag
(anr            INTEGER CONSTRAINT anr_primary PRIMARY KEY,
 knr            INTEGER CONSTRAINT a_knr_notnull NOT NULL,
 konr           INTEGER,
 adatum        DATE DEFAULT CURRENT_DATE,
 atext         CHAR(30),
 fertigist     DATE,
 fertigso11    DATE,
 astat         INTEGER DEFAULT 1 CONSTRAINT astat_notnull NOT NULL,
 CONSTRAINT    a_knr_ref_kunde FOREIGN KEY (knr) REFERENCES kunde,
 CONSTRAINT    konr_ref_kontakt FOREIGN KEY (konr) REFERENCES kontakt,
 CONSTRAINT    astat_ref_aufstat FOREIGN KEY (astat) REFERENCES
 aufstat(astnr)
)
```

Die Tabelle AUFTRAG enthält folgende Daten:

anr	knr	konr	adatum	atext	fertigist	fertigso11	astat
200	102	25	1990-04-10	Mitarbeiterschulung	1990-05-01	1990-05-01	5
210	106	40	1990-12-13	Kunden-Verwaltung	1991-04-20	1991-04-01	3
211	106	41	1990-12-29	Datenbank-Entwurf Kunden	1991-04-10	1991-04-01	4
250	105	35	1991-01-17	Serienbrief-Einweisung		1991-03-01	2
251	105	35	1991-01-17	Kunden-Verwaltung		1991-05-01	2
300	101	20	1991-02-15	Netzwerk-Test/Vergleich			1
305	105	35	1991-05-01	Mitarbeiterschulung		1991-05-01	2

Tabelle 17: Daten der Tabelle AUFTRAG

### 10.2.1.4 Tabelle LEISTUNG

Die Tabelle LEISTUNG ist wie folgt definiert:

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

Die Tabelle LEISTUNG enthält folgende Daten:

Inr	anr	ldatum	ltext	leinheit	lanz	lsatz	mwsatz	rnr
1	200	1990-04-20	Schulungsmaterial	Seiten	45	75	0.15	3
2	200	1990-04-25	Schulung	Tag	1	1500	0.15	3
3	200	1990-04-26	Schulung	Tag	1	1500	0.15	3
4	211	1991-01-20	Systemanalyse	Tag	8	1200	0.00	10
5	211	1991-01-29	Datenbankentwurf	Tag	10	1200	0.00	10
6	211	1991-02-15	Kopien/Folien	Seiten	30	50	0.15	10
7	211	1991-03-27	Handbuch	Festpreis	1	200	0.07	10
10	250	1991-02-20	Reisekosten	Festpreis	2	125	0.00	
11	250	1991-02-20	Schulung	Tag	0.5	1200	0.15	

Tabelle 18: Daten der Tabelle LEISTUNG

### 10.2.1.5 Tabelle AUFSTAT

Die Tabelle AUFSTAT ist wie folgt definiert:

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

Die Tabelle AUFSTAT enthält folgende Daten:

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

Tabelle 19: Daten der Tabelle AUFSTAT

## 10.2.2 Schema TEILE

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

### 10.2.2.1 Tabelle ARTIKEL

Die Tabelle ARTIKEL ist wie folgt definiert:

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

Die Tabelle ARTIKEL enthält folgende Daten:

artnr	artbez	farbe	preis	bestand	minbestand
1	Fahrrad	schwarz	700.50	2	1
10	Rahmen	schwarz	150.00	10	5
11	Rahmen	alpinweiss	150.00	10	5
120	Vorderrad	metallic	40.00	3	5
130	Hinterrad	metallic	40.00	12	5
200	Lenkstange	metallic	60.00	1	5
210	V-Nabe	metallic	5.00	15	1
220	H-Nabe	metallic	5.00	14	10
230	Felge	schwarz	10.00	9	10
240	Speiche	schwarz	1.00	211	00
500	Schraube M5	schwarz	1.10	300	00
501	Mutter M5	schwarz	0.75	295	00

Tabelle 20: Daten der Tabelle ARTIKEL

### 10.2.2.2 Tabelle VERWENDUNG

Die Tabelle VERWENDUNG ist wie folgt definiert:

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

Die Tabelle VERWENDUNG enthält folgende Daten:

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

Tabelle 21: Daten der Tabelle VERWENDUNG

### 10.2.2.3 Tabelle LAGER

Die Tabelle LAGER ist wie folgt definiert:

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

Die Tabelle LAGER enthält folgende Daten:

artnr	bestand	ort
1	2	Hauptlager
10	10	Hauptlager
11	10	Hauptlager
120	3	Hauptlager
130	3	Hauptlager
130	9	Teilelager
240	11	Hauptlager
240	200	Teilelager
500	120	Hauptlager
500	180	Teilelager

Tabelle 22: Daten der Tabelle LAGER

#### 10.2.2.4 Tabelle FARBTAB

Die Tabelle FARBTAB ist wie folgt definiert:

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

Die Tabelle FARBTAB enthält folgende Daten:

<b>farbname</b>	<b>rgb</b>		
feuerrot	0.98	0	0
orange	0.9	0.3	0
himmelblau	0	0	0.99
aquamarinblau	0	0.1	0.99
alpinweiss	0.99	0.99	0.99
schwarz	0	0	0
metallic	0	0.2	0.3

Tabelle 23: Daten der Tabelle FARBTAB

### 10.2.2.5 Tabelle TABTAB

Die Tabelle TABTAB ist wie folgt definiert:

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

Die Tabelle TABTAB enthält folgende Daten:

<b>tablenr</b>	<b>tablename</b>	<b>bemerkung</b>
1	artikel	Teiledaten
2	verwendung	Beziehungsdaten Fahrrad
3	lager	Lagerorte der Teile
4	farbtap	Erlaubte Farben
5	tabtab	Verwendete Tabellen

Tabelle 24: Daten der Tabelle TABTAB

---

# Fachwörter

Dieses Fachwortverzeichnis enthält Definitionen wichtiger Begriffe, die in diesem Handbuch verwendet werden.

*Kursiv* gedruckte Fachwörter in den Definitionen verweisen auf entsprechende Definitionen für diese Fachwörter.

In der Zeile „Synonym(e):“ werden bedeutungsgleiche oder bedeutungsähnliche Bezeichnungen genannt, die in der Literatur, nicht jedoch in diesem Handbuch oder in den übrigen SESAM/SQL-Handbüchern verwendet werden.

## **Basistabelle**

base table

Mit einer CREATE TABLE-Anweisung erstellte Tabelle, die permanent in einer Datenbank gespeichert wird. Im Rahmen der Migration werden ebenfalls Basistabellen erzeugt.

Basistabellen können von unterschiedlicher Tabellenart sein.

Anzahl und Datentyp der Spalten sowie Integritätsbedingungen werden mit der *SQL*-Anweisung CREATE TABLE vereinbart und können mit ALTER TABLE modifiziert werden. Die Anzahl der Zeilen ist durch die Tabellendefinition nicht festgelegt.

## **Benutzervariable**

host variable

Variable einer Wirtssprache (z.B. C, COBOL), die innerhalb einer *eingebetteten SQL-Anweisung* angesprochen wird. Benutzervariablen sind innerhalb von SQL-Anweisungen durch einen vorangestellten Doppelpunkt gekennzeichnet und müssen innerhalb der DECLARE-Section deklariert werden.

## **Cursor**

cursor

Zeiger innerhalb einer besonderen Ergebnistabelle, der sogenannten Cursortabelle, mit dem man auf die Sätze der Tabelle einzeln zugreifen kann.

Bei der Deklaration des Cursors wird der Cursorname vergeben. In der Cursorbeschreibung wird die Cursortabelle spezifiziert und es wird festgelegt, ob der Cursor änderbar sein soll und die Sätze in der Cursortabelle eine bestimmte Reihenfolge haben sollen.

**Data Base Handler (DBH)**

database handler (DBH)

Komponente von SESAM/SQL, die alle Datenbankzugriffe einer DBH-Session analysiert, ausführt und koordiniert.

Der Data Base Handler (DBH) ist in zwei Varianten einsetzbar:

- independent DBH  
Data Base Handler, der als selbständiges Programmsystem den Mehrbenutzerbetrieb unterstützt. Der independent DBH läuft in einer eigenen Task ab.
- linked-in DBH  
Data Base Handler, der exklusiv die Aufträge eines einzigen Anwenderprogramms bearbeitet und zum Anwenderprogramm gebunden wird. Der linked-in DBH läuft in derselben Task wie das Anwenderprogramm ab.

Synonym: SESAM/SQL-DBH, DBH

**Datenbank**

database

Zusammengehörige Datenbestände, die mit Hilfe eines Datenbanksystems ausgewertet, bearbeitet und verwaltet werden.

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

**dynamisch übersetzbare SQL-Anweisung**

preparable SQL statement

*SQL*-Anweisung, die erst zur Laufzeit des Programms in der Wirtssprache (z.B. C, COBOL) übersetzt wird. Auf diese Weise können beispielsweise Datenbankabfragen formuliert werden, die bei der Erstellung eines Programms noch nicht bekannt sind.

**eingebettete SQL-Anweisung**

embedded SQL statement

*SQL*-Anweisung, die innerhalb eines Programms der Wirtssprache (z.B. COBOL) angegeben wird und deren Anfang (mit EXEC SQL) und Ende (mit END-EXEC) markiert ist. Diese Markierungen heben die SQL-Anweisungen von den Anweisungen der Wirtssprache ab und ermöglichen die Precompilierung der SQL-Anweisungen.

**Indikatorvariable**

indicator variable

spezielle *Benutzervariable* vom ganzzahligen numerischen Datentyp SMALLINT, die den Inhalt einer anderen Benutzervariablen beschreibt. Die Indikatorvariable zeigt an, ob die zugeordnete Benutzervariable den NULL-Wert enthält, oder ob bei der Übertragung von alphanumerischen Werten in die Benutzervariable Datenverlust aufgetreten ist.

**multiple Spalte**

multiple column

*Spalte*, in der pro Datensatz mehrere Werte desselben Datentyps abgespeichert werden können. Einen solchen Wert nennt man Ausprägung einer multiplen Spalte.

Synonyme: multiples Attribut, multiples Feld

**Satz**

row (SQL)

Geordnete Folge von Werten, die einer *Zeile* einer *Tabelle* entspricht.

Synonym: *Tupel*, *Zeile*

**Schema**

schema

Schemata sind im Catalog Space der *Datenbank* enthalten.

Man unterscheidet die anwenderdefinierten Schemata und die Informationsschemata.

Anwenderdefinierte Schemata enthalten Metadaten, die den formalen Aufbau der *Basistabellen* und *Views* einer Datenbank beschreiben. Außerdem enthalten die anwenderdefinierten Schemata weitere Metadaten, wie z.B. Angaben zu Privilegien.

Ein anwenderdefiniertes Schema hat einen Namen und einen Eigentümer, der durch einen Berechtigungsschlüssel für dieses Schema ausgewiesen ist. Es wird mit der SQL-Anweisung CREATE SCHEMA erstellt und kann durch weitere Anweisungen (z.B. CREATE TABLE) modifiziert werden.

Bei den Informationsschemata unterscheidet man INFORMATION\_SCHEMA und SYS\_INFO\_SCHEMA, die für jede Datenbank vorhanden sind. Sie ermöglichen dem Anwender Zugriff auf die in den anwenderdefinierten Schemata abgelegten Metadaten, wie z.B. Beschreibungen von Basistabellen, Views, Integritätsbedingungen, Privilegien, usw.

Das INFORMATION\_SCHEMA kann jeder Anwender mit SQL-Mitteln abfragen. Das SYS\_INFO\_SCHEMA enthält systemspezifische Daten und ist nur dem universellen Benutzer zugänglich.

**Spalte**  
column

Bestandteil einer *Tabelle*. Jede Spalte besitzt einen Namen und einen Datentyp und enthält Spaltenwerte dieses Datentyps. Man unterscheidet einfache und *multiple Spalten*.

Synonym: Attribut

**SQL**  
SQL

(Structured Query Language) ist die am weitesten verbreitete Sprache zur Bearbeitung von relationalen Datenbanken. Im Gegensatz zu den prozeduralen Sprachen nicht-relationaler Datenbanksysteme ist SQL deskriptiv, d.h. der Anwender beschreibt in einer mengenorientierten Form das Ergebnis einer Datenbankoperation und nicht die zu diesem Ergebnis führenden Schritte. Dabei lehnt sich SQL an die englische Sprache an. SQL bietet umfangreiche Sprachmittel zur Datendefinition, Datenmanipulation, Transaktionsverwaltung, Verwaltung von Zugriffsrechten usw. sowie als Embedded SQL (ESQL) die Möglichkeit, innerhalb einer Wirtssprache (z.B. C, COBOL) mit *eingebetteten SQL-Anweisungen* auf eine Datenbank zuzugreifen. SQL wurde erstmals 1987 von der International Organization for Standardization normiert. SESAM/SQL basiert auf dem aktuellen internationalen Standard ISO/IEC 9075:2003, in diesem Handbuch kurz SQL2- Norm genannt.

**Tabelle**  
table

Eine Tabelle ist eine zweidimensionale Anordnung von Datenelementen, die aus Zeilen (horizontal) und *Spalten* (vertikal) besteht.

Man unterscheidet *Basistabellen*, *Views* und Ergebnistabellen.

Synonym: Relation

**Transaktion**

## transaction

Folge von zusammengehörigen Anweisungen (jeweils nur SQL- oder nur CALL-DML-Anweisungen), die eine *Datenbank* von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt. Eine Transaktion wird entweder vollständig oder überhaupt nicht ausgeführt.

Für die Transaktionsverarbeitung gibt es spezielle Anweisungen:

CALL-DML kennt jeweils eine Anweisung zum Öffnen, Schließen (Festschreiben) und Rücksetzen einer Transaktion.

In *SQL* gibt es spezielle Anweisungen nur zum Beenden und Rücksetzen einer Transaktion; in UTM-Anwendungen müssen für diesen Zweck die entsprechenden UTM-Aufrufe verwendet werden. Der Beginn einer Transaktion wird nicht mit einer speziellen SQL-Anweisung festgelegt: die erste transaktionseinleitende Anweisung nach dem Rücksetzen oder Festschreiben der vorherigen Transaktion oder dem Programmstart wertet SESAM/SQL als Transaktionsbeginn.

**View**

## view

Aus einer oder mehreren *Tabellen* abgeleitete benannte Tabelle. Ein View wird mit den *SQL*-Anweisungen CREATE VIEW oder CREATE TEMPORARY VIEW (temporärer View) durch einen Abfrage-Ausdruck definiert. Die durch den Abfrage-Ausdruck spezifizierte Ergebnistabelle wird jedesmal neu ermittelt, wenn der View in einer SQL-Anweisung angesprochen wird.

Somit enthält die Ergebnistabelle stets die aktuellen Werte der Datenbank.

Synonym: Benutzersicht



---

# Literatur

Die Handbücher sind online unter <http://manuals.fujitsu-siemens.com> zu finden oder in gedruckter Form gegen gesondertes Entgelt unter <http://FSC-manualshop.com> zu bestellen.

[1] **SESAM/SQL-Server** (BS2000/OSD)

Basishandbuch  
Benutzerhandbuch

*Zielgruppe*

Das Handbuch wendet sich an alle Anwender und alle, die sich über SESAM/SQL informieren wollen.

*Inhalt*

Das Handbuch gibt einen Überblick über das Datenbanksystem und beschreibt Grundlagen, Konzepte und Zusammenhänge. Es ist die Basis für das Verständnis der anderen SESAM/SQL-Handbücher.

[2] **SESAM/SQL-Server** (BS2000/OSD)

SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen  
Benutzerhandbuch

*Zielgruppe*

Zur Zielgruppe gehören alle Personen, die eine SESAM/SQL-Datenbank mit SQL-Anweisungen bearbeiten.

*Inhalt*

Das Handbuch beschreibt die Programmeinbettung von SQL-Anweisungen und die SQL-Sprachelemente. In einem alphabetischen Nachschlageteil sind alle SQL-Anweisungen ausführlich dargestellt.

[3] **SESAM/SQL-Server** (BS2000/OSD)

SQL-Sprachbeschreibung Teil 2: Utilities  
Benutzerhandbuch

*Zielgruppe*

Zur Zielgruppe gehören alle Personen, die mit der Verwaltung einer SESAM/SQL-Datenbank befasst sind.

*Inhalt*

Das Handbuch enthält eine alphabetische Beschreibung der Utility-Anweisungen; Utility-Anweisungen sind Anweisungen in SQL-Syntax und realisieren die Utility-Funktionen von SESAM/SQL.

[4] **SESAM/SQL-Server** (BS2000/OSD)

CALL-DML Anwendungen  
Benutzerhandbuch

*Zielgruppe*

Programmierer von SESAM-Anwendungen

*Inhalt*

- CALL-DML-Anweisungen für die Bearbeitung von SESAM-Datenbanken mittels Anwenderprogrammen
- Teilhaberbetrieb mit UTM und DCAM
- Dienstprogramme SEDI61 und SEDI63 zur Datenwiedergewinnung und Direktänderung
- Hinweise zum Mischbetrieb von CALL-DML- und SQL-Modus

[5] **SESAM/SQL-Server** (BS2000/OSD)

Datenbankbetrieb  
Benutzerhandbuch

*Zielgruppe*

Das Handbuch wendet sich an den SESAM/SQL-Systemverwalter.

*Inhalt*

Das Handbuch beschreibt, welche Möglichkeiten der Systemverwalter hat, den Datenbankbetrieb zu steuern und zu überwachen.

[6] **SESAM/SQL-Server** (BS2000/OSD)

Utility-Monitor  
Benutzerhandbuch

*Zielgruppe*

Das Handbuch ist für den Datenbankverwalter und den Systemverwalter von SESAM/SQL-Server bestimmt.

*Inhalt*

Das Handbuch beschreibt die Bedienung des Utility-Monitors. Mit dem Utility-Monitor können DB-Verwaltungs- und Administrationsaufgaben u.a. im maskengeführten Dialog ausgeführt werden.

[7] **SESAM/SQL-Server** (BS2000/OSD)

**Meldungen**

Benutzerhandbuch

*Zielgruppe*

Zur Zielgruppe gehören alle SESAM/SQL-Anwender.

*Inhalt*

Das Meldungshandbuch enthält Informationen über den Aufbau und Abruf der Meldungen des Datenbanksystems SESAM/SQL-Server und der Verteilkomponente SESAM/SQL-DCN. Die SQLSTATEs und CALL-DML-Statusmeldungen sind hier vollständig aufgeführt.

- [8] **SESAM/SQL-Server** (BS2000/OSD)  
Umstellen von SESAM-Datenbanken u. -Anwendungen auf SESAM/SQL-Server  
Benutzerhandbuch

*Zielgruppe*

Das Handbuch richtet sich an alle, die sich für SESAM/SQL-Server ab der Version 2.0 interessieren.

*Inhalt*

Dieses Handbuch beschreibt die neuen Konzepte und Funktionen im Überblick. Im Vordergrund steht der Bezug zu Vorgängerversionen, um bisherigen SESAM/SQL-Anwendern den Umstieg in die neue Welt von SESAM/SQL-Server zu erleichtern.

- [9] **SESAM/SQL-Server** (BS2000/OSD)  
Performance  
Benutzerhandbuch

*Zielgruppe*

Das Handbuch wendet sich an den erfahrenen Anwender von SESAM/SQL.

*Inhalt*

Das Handbuch beschreibt, wie man Performance-Engpässe im Leistungsverhalten von SESAM/SQL erkennt. Es enthält Maßnahmen, die geeignet sind, das Systemverhalten zu beeinflussen.

- [10] **openUTM V5.2**  
**Konzepte und Funktionen**  
Benutzerhandbuch

*Zielgruppe*

Alle, die sich über die Funktionsbreite und Leistungsfähigkeit von openUTM informieren wollen.

*Inhalt*

Das Handbuch enthält eine allgemeine Beschreibung aller Funktionen und Leistungen von openUTM sowie einführende Informationen, die als Einstieg in die Arbeit mit openUTM dienen sollen.

- [11] **openUTM (BS2000/OSD)**  
**Anwendungen generieren und betreiben**  
Benutzerhandbuch
- Zielgruppe*  
Das Handbuch richtet sich an Anwendungsplaner, Fachprogrammierer, Administratoren und Anwender von UTM-Anwendungen.
- Inhalt*  
Das Handbuch beschreibt die Generierung von UTM-Anwendungen mit verteilter Verarbeitung, die Tools, die openUTM dazu zur Verfügung stellt und die UTM-Objekte, die bei der Generierung erzeugt werden. Außerdem enthält das Handbuch alle Informationen, die für die Strukturierung, den Betrieb und die Kontrolle einer UTM-Produktivanwendung benötigt werden.
- [12] **UTM (TRANSDATA)**  
**Anwendungen programmieren**  
Benutzerhandbuch
- Zielgruppe*  
Programmierer von UTM-Anwendungen
- Inhalt*
- Sprachunabhängige Beschreibung der Programmschnittstelle KDCS,
  - Aufbau von UTM-Programmen
  - KDCS-Aufrufe
  - Testen von UTM-Anwendungen
  - Alle Informationen, die der Programmierer von UTM-Anwendungen benötigt
- Einsatz*  
BS2000-Transaktionsbetrieb
- [13] **openUTM V3.4A, openUTM-D V3.4A (BS2000/OSD)**  
Neue Schnittstellen und Funktionen  
Ergänzungen zu den UTM-Handbüchern der V3.3A/3.4A  
Benutzerhandbuch
- Zielgruppe*  
Organisierer, Einsatzplaner, Programmierer und Administratoren von UTM-Anwendungen.
- Inhalt*  
Ergänzungshandbuch zu den Handbüchern zu UTM V3.3A/UTM-D V2.0A. Es enthält die Beschreibung der X/Open-Schnittstellen CPI-C und XATMI unter UTM und der in UTM V3.3B und UTM V3.4 neuen Funktionen.

- [14] **FHS (TRANSDATA)**  
Benutzerhandbuch  
*Zielgruppe*  
Programmierer  
*Inhalt*  
Programmschnittstellen von FHS für TIAM-, DCAM- und UTM-Anwendungen. Erstellen, Einsatz und Verwalten von Formaten.
- [15] **CRTE V2.5A (BS2000/OSD)**  
Common Runtime Environment  
Benutzerhandbuch  
*Zielgruppe*  
Programmierer und Systemverwalter im BS2000/OSD  
*Inhalt*  
Beschreibung der gemeinsamen Laufzeitumgebung für COBOL85-, COBOL2000-, C-, und C++-Objekte sowie für "Fremdsprachenmix":  
- Komponenten des CRTE  
- Programmkommunikationsschnittstelle ILCS  
- Bindebeispiele
- [16] **COBOL2000 (BS2000/OSD))**  
**COBOL-Compiler**  
Sprachbeschreibung  
*Zielgruppe*  
COBOL-Anwender im BS2000/OSD  
*Inhalt*  
– COBOL-Glossary  
– Einführung in Standard-COBOL  
– Beschreibung des gesamten Sprachumfangs des COBOL2000-Compilers:  
Formate, Regeln und Beispiele zu den COBOL-ANS'85-Sprachelementen der Sprachmenge "High" , den Fujitsu Siemens-spezifischen Spracherweiterungen sowie den Erweiterungen des kommenden COBOL-Standards, insbesondere der Objektorientierung.

[17] **COBOL2000** (BS2000/OSD))

**COBOL-Compiler**

Benutzerhandbuch

*Zielgruppe*

COBOL-Anwender im BS2000/OSD

*Inhalt*

- Bedienung des COBOL2000-Compilers
- Binden, Laden und Starten von COBOL-Programmen
- Testhilfen
- Dateiverarbeitung mit COBOL-Programmen
- Fixpunktausgabe und Wiederanlauf
- Programmverknüpfung
- COBOL2000 und POSIX-Subsystem
- Nutzbare Software für COBOL-Anwender
- Meldungen des COBOL2000-Systems

[18] **BS2000/OSD-BC**

**Kommandos Band 1 - 5**

Benutzerhandbuch

*Zielgruppe*

Das Handbuch wendet sich sowohl an den nichtprivilegierten Anwender als auch an die Systembetreuung.

*Inhalt*

Die Bände 1 - 5 enthalten die Kommandos ADD-... bis WRITE-... (BS2000/OSD-Grundausbau und ausgewählte Produkte) mit der Funktionalität für alle Privilegien. Die Kommando- und Operandenfunktionen werden ausführlich beschrieben; viele Beispiele unterstützen das Verständnis. Am Anfang jedes Bandes informiert eine Übersicht über alle in den Bänden 1 - 5 beschriebenen Kommandos.

Der Anhang von Band 1 enthält u.a. Informationen zur Kommandoeingabe, zu bedingten Jobvariablenausdrücken, Systemdateien, Auftragsschaltern, Geräte- und Volumetypen. Der Anhang der Bände 4 und 5 enthält jeweils eine Übersicht zu den Ausgabespalten der SHOW-Kommandos der Komponente NDM. Der Anhang von Band 5 enthält zusätzlich eine Übersicht aller START-Kommandos.

In jedem Band ist ein umfangreiches Stichwortverzeichnis mit allen Stichwörtern der Bände 1 - 5 enthalten.

- [19] **BINDER**  
**Binder in BS2000/OSD**  
Benutzerhandbuch
- Zielgruppe*  
Das Handbuch wendet sich an Software-Entwickler
- Inhalt*  
Es beschreibt die BINDER-Funktionen und enthält Beispiele dazu. Im Nachschlageteil sind die BINDER-Anweisungen und der Makroaufruf BINDER beschrieben.
- [20] **BLSSERV**  
**Bindelader-Starter in BS2000/OSD**  
Benutzerhandbuch
- Zielgruppe*  
Das Handbuch wendet sich an Software-Entwickler und geübte BS2000/OSD-Benutzer.
- Inhalt*  
Es beschreibt die Funktionen, die Unterprogrammchnittstelle, die XS-Unterstützung und den Aufruf des Bindeladers DBL als Bestandteil des Subsystems BLSSERV.

## Sonstige Literatur

**International Organization for Standardization (ISO):**  
**Database Language SQL**  
**ISO/IEC 9075:2003**  
Kurztitel: SQL2-Norm

**An introduction to Database Systems,**  
C.J.Date  
Addison Wesley, 2003



---

# Stichwörter

## A

Aliasname 15  
alphanum-name (Datentyp) 16  
alphanumerisches Datenfeld 32  
Änderungen (Funktionen) 10  
Anweisung  
    Syntaxdarstellung 12  
Aufbau des Kommunikationsbereichs 69  
Aufrufen des ESQL-Precompiler 80  
Ausführungseinheit aus mehreren ESQL-COBOL-  
    Anwendungen 112  
Ausgabeziel festlegen  
    für erzeugtes COBOL-Programm 87  
    für SQL-Bindelad Modul 90  
Ausnahmebehandlung 75

## B

Basistabelle 171  
Begrenzer festlegen 100  
Beispiel  
    Programm ABFRAGE 120  
    Programm AENDERN 125  
    Programm DYNAMISCH 135  
    Programm EINFUEGEN 128  
    Programm LOESCHEN 131  
    Schema Auftragsverwaltung 160  
Beispieldatenbank 160  
Benutzervariable 30, 171  
    Benennung 32, 35  
    Datenname 32  
    Datentyp 30  
    definieren 30  
    erlaubte Datentypen 43–65  
    in SQL-Anweisung angeben 37  
    ISO-Sprachumfang 35

    Stufennummer 32

    Vektor 38

Berechtigungsschlüssel angeben 101  
Bereitstellen des Kommunikationsbereichs 72  
Binden 29, 111  
BINDER 111

## C

c-string (Datentyp) 16  
CALL-DML-Schnittstelle 160  
cat (Zusatz zu Datentypen) 26  
cat-id (Datentyp) 16  
Catalogname voreinstellen 100  
CHARACTER (Datentyp) 44  
COBOL-Bindemodul einbinden 111  
COBOL-Datentypen  
    Zuordnung zu SQL-Datentypen 43–65  
command-rest (Datentyp) 16  
compl (Zusatz zu Datentypen) 21  
composed-name (Datentyp) 16  
constr (Zusatz zu Datentypen) 25  
corr (Zusatz zu Datentypen) 26  
CRTE einbinden 112  
Cursor 171

## D

Darstellungsmittel 11  
Data Base Handler 172  
DATE (Datentyp) 34, 59  
date (Datentyp) 16  
Datenbank 172  
Datenbankkontakt 93  
Datenbankname  
    voreinstellen 100  
Datennamen für Benutzervariablen 32

## Datentypen

- CHARACTER 44
- CHARACTER VARYING 45
- DATE 59
- DECIMAL 54
- DOUBLE PRECISION 58
- FLOAT 58
- INTEGER 52
- NCHAR 47
- NUMERIC 55
- NVARCHAR 48
- REAL 57
- SMALLINT 50
- TIME(3) 61
- TIMESTAMP(3) 63
- VARCHAR 45
- Vektoren 65

Datentypen für Benutzervariablen 30, 43–65

Datentypen SDF 12, 16

- Zusätze 13

Datum (Datentyp) 59

DBL 111

DECIMAL (Datentyp) 54

DECLARE SECTION 30

- Kommentare 31

device (Datentyp) 16

Diagnoseunterlagen erstellen 108

DOUBLE PRECISION (Datentyp) 58

dynamisch übersetzbare SQL-Anweisung 172

## E

Eigenschaften des ESQL-COBOL-Programms

- festlegen 98

Eingabequelle für den ESQL-Precompiler

- angeben 96

eingebettete SQL-Anweisung 172

Einsprungrname des SQL-Bindeladmoduls 94

Erfolgskontrolle 73

Ersetzungen durchführen 103

Ersetzungsdatei angeben 101

ESQL-COBOL-Anwendung

- als UTM-Teilprogramm 116
- binden 111
- starten 113

ESQL-COBOL-Programm

- Eigenschaften festlegen 98
- Informationsteil 107
- übersetzen 109

ESQL-Precompiler

- aufrufen 80
- Beendungsverhalten 105
- Eingabequelle angeben 96
- Meldungen 106
- mit Jobvariable überwachen 81
- steuern 84

ESQL-Precompiler-Optionen 84–104

- Überblick 84

EXTERNAL-Klausel 33

## F

Fehlerbehandlung 73

Fehlerklasse 107

Festpunktzahl (Datentyp)

- gepackt 54
- ungepackt 55

FHS-Startparameter angeben 118

fixed (Datentyp) 16

FLOAT (Datentyp) 58

full-filename (Datentyp) 17

## G

Ganzzahl (Datentyp) 52

- kleine 50

gen (Zusatz zu Datentypen) 26

gepackte Festpunktzahl (Datentyp) 54

geschachteltes ESQL-COBOL-Programm 35

Gleitpunktzahl (Datentyp) 58

- doppelte Genauigkeit 58
- einfache Genauigkeit 57

GLOBAL-Klausel 33

GROUP-USAGE-Klausel 33

## H

HOST-LANGUAGE (Precompiler-Option) 98

HOST-PROGRAM (Precompiler-Option) 87

**I**

INCLUDE (SQL-Anweisung) 77  
 INCLUDE-Bibliothek angeben 89  
 INCLUDE-Elemente 77  
 INCLUDE-LIBRARY (Precompiler-Option) 89  
 independent DBH 94

## Index

-Schreibweise 25  
 global 24  
 Konstruktionszeichenfolge 24  
 platzhalter-spezifisch 24

INDICATOR (Schlüsselwort) 41

Indikatorvariable 40, 173

abgelegte Werte 42  
 definieren 40  
 in einer SQL-Anweisung angeben 41  
 NULL-Wert übertragen 42

INTEGER (Datentyp) 52

integer (SDF-Syntax) 18

ISO-Sprachumfang festlegen 99

**J**

Jobvariable angeben 92

**K**

KDCDEF-Anweisung DATABASE 116

KDCROOT 116

kleine Ganzzahl (Datentyp) 50

Kommando

Syntaxdarstellung 12

Kommentar

in einer DECLARE SECTION 31  
 in einer SQL-Anweisung 68

Kommunikationsbereich 69

Aubau 69  
 bereitstellen 72  
 für ESQL-COBOL V1.1-Anwendungen 69  
 Varianten 69

Konfigurationsdatei

zuweisen 113

Konstruktionsangabe 25

Konstruktionszeichenfolge 24

Kurzname 15

**L**

linked in-DBH 94

Literaturverweise 9

low (Zusatz zu Datentypen) 21

**M**

man (Zusatz zu Datentypen) 26

Maskenzeichen 32, 43

Meldungen 106, 141

Aufbau 106

Metasprache siehe [Darstellungsmittel](#)

Metasyntax SDF 12, 14

Mischbetrieb 160

MODULE-LIBRARY (Precompiler-Option) 90

MONJV (Precompiler-Option) 92

multiple Spalte 173

**N**

name (Datentyp) 18

nationale Zeichenkette fester Länge  
 (Datentyp) 47

nationale Zeichenkette variabler Länge  
 (Datentyp) 48

NCHAR (Datentyp) 47

Neuerungen (Funktionen) 10

NULL-Wert übertragen 42

NUMERIC (Datentyp) 55

numerisches Datenfeld 32

NVARCHAR (Datentyp) 33, 48

**O**

OCCURS-Klausel 33

odd (Zusatz zu Datentypen) 26

Optionen des ESQL-Precompiler 84–104

**P**

partial-filename (Datentyp) 19

PICTURE-Klausel 32

Portierung 99

Ersetzungen durchführen 103

posix-filename (Datentyp) 19

posix-pathname (Datentyp) 19

POSIX-Platzhalter 22

PRECOMPILER-ACTION (Precompiler-Option) [93](#)  
Precompiler-Optionen [84–104](#)  
  HOST-PROGRAM [87](#)  
  INCLUDE-LIBRARY [89](#)  
  MODULE-LIBRARY [90](#)  
  MONJV [92](#)  
  PRECOMPILER-ACTION [93](#)  
  SOURCE [96](#)  
  SOURCE-PROPERTIES [98](#)  
  Überblick [84](#)  
product-version (Datentyp) [20](#)

**Q**  
Querverweisliste erzeugen [95](#)

**R**  
Readme-Datei [9](#)  
REAL (Datentyp) [57](#)  
RUN UNIT [112](#)

**S**  
Satz [173](#)  
Schema [173](#)  
SCHEMA voreinstellen [100](#)  
SDF  
  Syntaxdarstellung [12](#)  
sep (Zusatz zu Datentypen) [26](#)  
SESAM-Makrobibliothek [116](#)  
SESAM/SQL-Konnektionsmodul  
  einbinden [112](#)  
SESAM/SQL-Modulbibliothek  
  zuweisen [113, 118](#)  
SIGN-Klausel [33](#)  
SMALLINT (Datentyp) [50](#)  
SOURCE (Precompiler-Option) [96](#)  
SOURCE-PROPERTIES (Precompiler-Option) [98](#)  
  im ESQL-COBOL-Programm angeben [80](#)  
Spalte [174](#)  
  multiple [173](#)  
Sprachumfang  
  festlegen [99](#)  
  unter UTM [115](#)

SQL [8, 174](#)  
SQL-Anweisung  
  Benutzervariable angeben [37](#)  
  dynamisch übersetzbar [172](#)  
  eingebettet [172](#)  
  in einem COBOL-Programm angeben [66](#)  
  INCLUDE [77](#)  
  Indikatorvariable angeben [41](#)  
  WHENEVER [75](#)  
SQL-Bindelademodul  
  einbinden [112](#)  
  Einsprunghname [94](#)  
SQL-Datentypen  
  Zuordnung zu COBOL-Datentypen [43–65](#)  
SQL-Kommentar [68](#)  
SQLca (Variable) [70](#)  
SQLCODE (Variable) [71](#)  
  definieren [72](#)  
SQLda (Variable) [71](#)  
SQLerrcol (Variable) [71](#)  
SQLerrline (Variable) [71](#)  
SQLerrm (Variable) [71](#)  
SQLline (Variable) [71](#)  
SQLrowcount (Variable) [72](#)  
SQLSTATE (Variable)  
  definieren [72](#)  
START-ESQLCOB (Kommando) [81](#)  
START-PROGRAM (Kommando) [82](#)  
Starten des ESQL-Precompiler [80](#)  
structured-name (Datentyp) [20](#)  
Stufennummern für Benutzervariablen [32](#)  
Syntaxdarstellung SDF [12](#)

**T**  
Tabelle [174](#)  
text (Datentyp) [20](#)  
time (Datentyp) [20](#)  
TIME(3) (Datentyp) [34, 61](#)  
TIMESTAMP(3) (Datentyp) [34, 63](#)  
Transaktion [175](#)

**U**  
Übersetzung [29, 109](#)  
Übertragung von Werten überprüfen [42](#)

Uhrzeit (Datentyp) 61  
under (Zusatz zu Datentypen) 21  
ungepackte Festpunktzahl (Datentyp) 55  
untergeordnetes Datenfeld  
  Namen qualifizieren 37  
USAGE-Klausel 32  
user (Zusatz zu Datentypen) 26  
UTM-Anwendung  
  generieren 116  
  Konfigurationsdatei 118  
  starten 118  
UTM-gerechter Sprachumfang 99  
UTM-Sprachumfang 115  
UTM-Startparameter angeben 118

**V**

VALUE-Klausel 33  
VARCHAR (Datentyp) 33, 45  
Varianten des Kommunikationsbereichs 69  
Vektor 38, 65  
vers (Zusatz zu Datentypen) 26  
View 175  
voreingestellt  
  Catalogname 100  
  Datenbankname 100  
  Schemaname 100  
Vorgängerversion 10  
Vorübersetzung 29, 81–108  
  Datenbankkontakt 93  
  Kommandofolge 83  
  starten 81  
  steuern 84, 93  
vsn (Datentyp) 20

**W**

WHENEVER (SQL-Anweisung) 75  
wild(n) (Zusatz zu Datentypen) 22  
with (Zusatz zu Datentypen) 21  
without (Zusatz zu Datentypen) 26

**X**

x-string (Datentyp) 21  
x-text (Datentyp) 21

**Z**

Zeichenkette fester Länge (Datentyp) 44  
Zeichenkette variabler Länge (Datentyp) 45  
Zeitstempel (Datentyp) 63  
Zusätze zu Datentypen 13, 21



Fujitsu Siemens Computers GmbH  
Handbuchredaktion  
81730 München

Kritik  
Anregungen  
Korrekturen

**Fax: 0 700 / 372 00001**

e-mail: [manuals@fujitsu-siemens.com](mailto:manuals@fujitsu-siemens.com)  
<http://manuals.fujitsu-siemens.com>

---

Absender

---

Kommentar zu ESQ-L-COBOL V3.0A  
ESQ-L-COBOL für SESAM/SQL-Server





## Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format ...@[ts.fujitsu.com](mailto:ts.fujitsu.com).

The Internet pages of Fujitsu Technology Solutions are available at [http://ts.fujitsu.com/...](http://ts.fujitsu.com/) and the user documentation at <http://manuals.ts.fujitsu.com>.

Copyright Fujitsu Technology Solutions, 2009

## Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form ...@[ts.fujitsu.com](mailto:ts.fujitsu.com).

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter [http://de.ts.fujitsu.com/...](http://de.ts.fujitsu.com/), und unter <http://manuals.ts.fujitsu.com> finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009