Einleitung

DRIVE/WINDOWS ermöglicht mit SQL-Anweisungen den Zugriff auf das Datenbanksystem SESAM/SQL. Über das Zusatzprodukt ISO/SQL erfüllt SESAM auch die SQL-Norm (ISO/IEC 9075:1989), bis auf geringfügige Einschränkungen (siehe Handbuch "SQL für ISO/SQL V1.0 Sprachbeschreibung" [11]).

Dieses Lexikon beschreibt die DRIVE-SQL-Anweisungen für SESAM in der DRIVE/WINDOWS-Version 1.1 für BS2000 und SINIX.

Dieses Lexikon beschreibt in Kurzform die Syntax der DRIVE-SQL-Anweisungen für ISO/SQL. Eine ausführliche Beschreibung der ISO/SQL-Anweisungen finden Sie darüber hinaus im Handbuch "SQL für ISO/SQL V1.0 Sprachbeschreibung" [11].

Komplexe Anweisungsteile, die sowohl in DRIVE- als auch in SQL-Anweisungen enthalten sind, werden gesondert beschrieben im Kapitel "Metavariablen" des Benutzerhandbuches "DRIVE/WINDOWS V 1.0: Lexikon der Anweisungen" [3]. Davon abweichende DRIVE-SQL-Metavariablen werden in diesem Lexikon in einem eigenen Kapitel "ISO/SQL-Metavariablen" beschrieben. Der Begriff "Betriebsmodus mit TP-Monitor" bezeichnet im BS2000 den UTM-Betrieb.

SQL-Returncodes werden von den Datenbanksystemen für SESAM/SQL, UDS/SQL und ISO/SQL übernommen und von DRIVE/WINDOWS als Fehlermeldung der Form DRI9xxx ausgegeben. Sie können nachgeschlagen werden im Handbuch "SQL für ISO/SQL (BS2000) V1.0: Sprachbeschreibung" [11].

ISO/SQL-Anweisungen

CLOSE Cursor schließen

CLOSE schließt einen Cursor, den Sie mit der Anweisung DECLARE für einen SELECT-Ausdruck vereinbart und mit OPEN geöffnet haben.

Vor einem erneuten Zugriff auf den Cursor mit der Anweisung FETCH müssen Sie den Cursor mit OPEN neu öffnen.

CLOSE ist sinnvoll, wenn bei der Deklaration eines Cursors bei DECLARE Variablen in *select-ausdruck* verwendet werden. Durch CLOSE und nachfolgend OPEN auf diesen Cursor wird *select-ausdruck* mit den zum OPEN-Zeitpunkt gültigen Werten der Variablen aktualisiert und die Cursortabelle mit den aktuellen Daten aus der Datenbank gefüllt.

Ein geöffneter Cursor wird auch bei Transaktionsende geschlossen.

Im Programm-Modus sind Cursor nur in der Quellprogrammdatei ansprechbar, in der sie mit DECLARE vereinbart werden.

```
CLOSE cursor
```

cursor Name des Cursors, den Sie schließen wollen.

COMMIT WORK Transaktion beenden

COMMIT WORK beendet eine Transaktion und schreibt die während der Transaktion durchgeführten Änderungen in der Datenbank fest.
Die erste fehlerfreie SQL-Anweisung nach COMMIT WORK eröffnet eine neue Transaktion.

COMMIT WORK schließt alle innerhalb der Transaktion geöffneten Cursor. Ein mit TEMPORARY definierter Cursor wird in einem DRIVE-Programm beim nächsten COMMIT WORK auf höherer Programmebene gelöscht.

Empfehlung:
Im DRIVE-Programm sollte unmittelbar nach den Deklarationen die Anweisung COMMIT WORK gegeben werden, um die Deklarationen in der Datenbank festzuschreiben.

Innerhalb einer Schleife mit Cursorverarbeitung ist COMMMIT WORK nur erlaubt, wenn der Cursor mit STORE gespeichert und mit RESTORE wiederhergestellt wurde (siehe Beispiel). Der Cursor muß dabei gemäß dem Handbuch "SQL für SESAM/SQL V1.0" [12] bzw. "SQL für UDS/SQL V1.0"[13] deklariert werden.

```
                  ⎡display     ⎤
COMMIT WORK [WITH ⎨send message⎬]
                  ⎣stop        ⎦
```

WITH — Mit WITH kann eine Anweisung festgelegt werden, die nach Transaktionsende ausgeführt wird.

Wird WITH in einem Dialog-Programm im Betriebsmodus mit TP-Monitor nicht angegeben, wird die Transaktion beendet und der Programmmlauf im Folge-Teilprogramm ohne Bildschirmausgabe fortgesetzt.

WITH darf nur im Programm-Modus angegeben werden.

display — Ausgeben eines Formats. Folgende Anweisungen dürfen verwendet werden:

- DISPLAY *screenformat* (siehe "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Anweisung DISPLAY *screenformat*)
- DISPLAY *formatname* (siehe "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Anweisung DISPLAY *formatname*)
- DISPLAY FORM (siehe "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Anweisung DISPLAY FORM)

send message
: Ausgeben von Meldungen (siehe "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Anweisung SEND MESSAGE).

stop
: Beenden des DRIVE-Laufs (siehe "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Anweisung STOP).

Regel

Wird die Transaktion nicht explizit vom Anwender mit COMMIT WORK, ROLLBACK WORK oder mit UTM-Sprachmitteln beendet, wird bei Ende der Ausführungseinheit (RUN UNIT) die Transaktion implizit zurückgesetzt. Der Anwender erhält eine Fehlermeldung von DRIVE/WINDOWS.

Beispiel für Cursorverarbeitung in einer Schleife:

```
DECLARE c1 ...        /*   gemäß Handbuch "SQL für SESAM/SQL V1.0" [12] bzw.
                           "SQL für UDS/SQL V1.0" [13]      */
  .
  .
CYCLE c1 INTO &var.*;
  .
  .
STORE c1;
COMMIT WORK;
RESTORE c1;
  .
  .
END CYCLE;
```

DECLARE... CURSOR FOR... Cursor deklarieren

DECLARE deklariert einen Cursor und weist dem Cursor ggf. eine Cursor-Spezifikation zu, die die Cursortabelle beschreibt. Die Deklaration muß im Deklarationsteil erfolgen. Sie ist innerhalb einer ganzen Übersetzungseinheit gültig.

Empfehlung:
Im DRIVE-Programm sollte unmittelbar nach den Deklarationen die Anweisung COMMIT WORK gegeben werden, um die Deklarationen in der Datenbank festzuschreiben.

Der Gültigkeitsbereich eines Cursors wird aufgehoben bei
- Programmende
- Programmabbruch
- DRIVE-Ende (STOP).
- DROP CURSOR *cursor* (DRIVE-Anweisung, im Dialog-Modus, dynamisch oder bei variablem Cursor)
- DROP CURSORS (DRIVE-Anweisung nur im Dialog-Modus oder dynamisch)

Gültigkeitsbereich eines Cursors:
Die DRIVE-Anweisungen DROP CURSOR *cursor* und DROP CURSORS sind nur im Dialog-Modus von DRIVE/WINDOWS oder als dynamische Anweisung erlaubt und löschen Cursor.
Beim Übergang vom Dialog- in den Programm-Modus von DRIVE/WINDOWS bleibt die Cursor-Definition erhalten, die Cursor-Position hingegen nicht.
Im Programm-Modus von DRIVE/WINDOWS bleibt ein Cursor, der mit PERMANENT definiert wurde, über das Ende eines mit CALL gerufenen Programms hinaus mit seiner Cursor-Position erhalten.
Der Gültigkeitsbereich des Cursors im Programm-Modus wird aufgehoben beim Übergang in den Dialog-Modus, bei Programmabbruch und DRIVE-Ende (STOP oder COMMIT WORK WITH STOP).
Ein Cursor, der mit TEMPORARY definiert wurde, wird beim nächsten COMMIT WORK auf höherer Programmebene gelöscht.

Mit dem Cursor können Sie auf die Sätze der Cursortabelle einzeln zugreifen. Der aktuelle Satz auf den der Cursor zeigt, kann gelesen, geändert oder gelöscht werden. Ändern oder Löschen in der Cursortabelle bedeutet immer auch ein Ändern oder Löschen in der zugrundeliegenden Basistabelle.

```
                    ┌──────────┐
DECLARE cursor [{   │PERMANENT │ }] CURSOR [PREFETCH n] [FOR cursor-spezifikation]
                    │TEMPORARY │
                    └──────────┘
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

cursor-spezifikation ::= union-ausdruck

                                                ┌─────────┐   ┌──────┐
                                    [ORDER BY {{│ganzzahl  │ }[ {│ASC   │ }]},...]
                                                │satzelement│  │DESC  │
                                                └─────────┘   └──────┘
```

cursor — Name des Cursors.
Alle Cursornamen müssen innerhalb einer Übersetzungseinheit eindeutig sein. Auf einer Programmebene bzw. im Dialog-Modus können zu einem Zeitpunkt keine zwei Cursor mit identischem Namen deklariert sein.

PERMANENT — Die Position des Cursors bleibt über das Ende des mit CALL gerufenen Programms hinaus erhalten, wenn weder im gerufenen Programm noch im rufenden Programm zwischen den CALL-Aufrufen ein COMMIT WORK durchlaufen wird. Die Cursor-Position wird nur aufgehoben, wenn der Cursor explizit mit CLOSE geschlossen wird bzw. beim Übergang in den Dialog-Modus.

> **i** Beim ersten Ablauf des mit CALL gerufenen Programms muß der Cursor mit der OPEN-Anweisung geöffnet werden, bei weiteren Abläufen darf keine OPEN-Anweisung mehr auf diesen Cursor gegeben werden. Im rufenden Programm darf keine COMMIT WORK-Anweisung stehen.

TEMPORARY — Der Cursor wird beim Ende des mit CALL gerufenen Programms geschlossen, seine Position aufgehoben (Gültigkeitsbereich beendet). Beim nächsten COMMIT WORK auf höherer Programmebene wird der Cursor gelöscht (Lebensdauer beendet).

PREFETCH — Die PREFETCH-Klausel ist bei ISO/SQL syntaktisch erlaubt.
Performancegewinne durch den Schubmodus ergeben sich bei Cursordeklarationen gemäß Handbuch "SQL für SESAM/SQL V1.0 Sprachbeschreibung" [12].
Folgende Anweisungen sind bei einem mit PREFETCH deklarierten Cursor nicht erlaubt:
FETCH PRIOR, FIRST, LAST (nur Positionieren mit FETCH NEXT erlaubt)
STORE und RESTORE
DELETE WHERE CURRENT OF
UPDATE WHERE CURRENT OF

n

n gibt die Anzahl der Sätze an, die bei der ersten FETCH-Anweisung vom Database Handler (DBH) in den DRIVE-Puffer eingelesen werden. Bei der 2. bis n-ten FETCH-Anweisung muß nicht auf die Datenbank zugegriffen werden. n muß größer oder gleich 2 sein. Ist l die Summe der Längen aller selektierten Satzelemente, dann sollte n * l kleiner oder gleich 30000 sein. Bei einer Bildschirmausgabe kann die Anzahl der Sätze, die auf einen Bildschirm ausgegeben werden können, als Richtlinie dienen.

FOR-Klausel

Die FOR-Klausel kann nur im Programm-Modus bei einer statischen Cursordeklaration weggelassen werden. Wird sie weggelassen, so wird ein sogenannter variabler Cursor deklariert. Ein solcher Cursor wird erst durch eine nachfolgende dynamische Deklaration mit FOR-Klausel gegenüber dem Datenbanksystem bekannt. Bei der dynamischen Deklaration mit EXEC müssen Sie die FOR-Klausel angeben. Bis auf die FOR-Klausel müssen die statische und die dynamische Deklaration eines Cursors gleich sein. Alle anderen Cursor-Anweisungen (OPEN, FETCH, DROP, CYCLE) können für den variablen Cursor statisch angegeben werden, wodurch die Performance steigt (siehe DRIVE-Programmiersprache [2] Kapitel Dynamische SQL-Anweisungen).

union-ausdruck

bestimmt die Cursortabelle durch Auswahl von Sätzen und Satzelementen aus bestehenden Basistabellen oder Views (siehe *union-ausdruck*).

ORDER BY sortiert die Sätze der Cursortabelle auf- oder absteigend nach den Werten der durch *satzelement* oder *ganzzahl* identifizierten Satzelemente.

Sind mehrere Satzelemente angegeben, wird zuerst nach den Werten des ersten angegebenen Satzelements sortiert. Wenn Sätze mit gleichen Werten für dieses Satzelement vorkommen, werden diese Sätze gemäß den Werten des zweiten Satzelements sortiert usw. bis entweder Eindeutigkeit vorliegt oder das letzte Satzelement erreicht ist. Bei der Sortierung mit der ORDER BY-Klausel wird ein NULL-Wert kleiner als Nicht-NULL-Werte interpretiert.

ORDER BY-Klausel nicht angegeben:
Die Cursortabelle ist unsortiert.

satzelement Name des Satzelements, nach dem sortiert werden soll.
Das Satzelement muß in *select-liste* von *union-ausdruck* vorkommen und darf nicht mit *tabelle* qualifiziert sein.

ganzzahl bezeichnet die Position des Satzelements in der Cursortabelle, nach dem sortiert werden soll (1 ≤ *ganzzahl* ≤ Anzahl der Satzelemente).
Die Satzelemente sind entsprechend den Angaben in *select-liste* von *union-ausdruck* von links nach rechts, beginnend mit 1, durchnumeriert.
Auf diese Weise können Sie auch Satzelemente als Sortierbegriff verwenden, die keinen Namen haben, wie z.B. Ergebnisfelder von Berechnungen.

ASC Die Werte der betreffenden Spalte werden aufsteigend sortiert. ASC ist Voreinstellung.

DESC Die Werte der betreffenden Spalte werden absteigend sortiert.

Änderbarer Cursor
Nur wenn ein Cursor änderbar ist, kann er zum Ändern oder Löschen mit den Anweisungen UPDATE... WHERE CURRENT OF... bzw. DELETE... WHERE CURRENT OF... verwendet werden. Ist die PREFETCH-Klausel angegeben, so kann auch ein änderbarer Cursor nicht zum Ändern oder Löschen verwendet werden.

Ein Cursor ist änderbar, wenn *union-ausdruck* änderbar ist (siehe *union-ausdruck*).

DELETE... WHERE bedingung Sätze löschen

DELETE... WHERE *bedingung* löscht Sätze aus einer Basistabelle. Diese Anweisung können Sie auch für einen änderbaren View angeben.

```
DELETE FROM tabelle [WHERE bedingung]
```

---

tabelle bezeichnet eine Basistabelle oder einen änderbaren View.

WHERE bedingung

Sätze, die die angegebene *bedingung* erfüllen, werden gelöscht.

Für jeden Datensatz wird geprüft, ob *bedingung* erfüllt ist.

Jede Unterabfrage (siehe *unterabfrage*) innerhalb von *bedingung* wird ausgewertet und das Ergebnis beim Abprüfen von *bedingung* verwendet.

Fehlt die WHERE-Klausel, so werden alle Sätze der angegebenen Basistabelle oder die mit dem View ausgewählten Sätze gelöscht.

DELETE... WHERE CURRENT OF... Aktuellen Satz des Cursors löschen

DELETE... WHERE CURRENT OF... löscht den aktuellen Satz des Cursors aus der zugrundeliegenden Basistabelle. Diese Anweisung können Sie auch für einen änderbaren View angeben.

Anschließend ist der Cursor vor dem nachfolgenden Satz positioniert. Erst durch eine nachfolgende FETCH-Anweisung wird der Cursor auf den nächsten Satz der Cursortabelle positioniert.

```
DELETE FROM tabelle WHERE CURRENT OF cursor
```

tabelle — Bezeichnet eine Basistabelle oder einen änderbaren View. Der angegebene Name einer Tabelle muß übereinstimmen mit dem Tabellennamen aus der FROM-Klausel der zugrundeliegenden Deklaration von *cursor*.

cursor — Name des Cursors.

Regeln

- Der Cursor muß vorher mit DECLARE deklariert worden sein.
- Der Cursor muß mit OPEN geöffnet und mit FETCH auf einen Satz positioniert sein.
- Der Cursor muß änderbar sein (siehe DECLARE...CURSOR FOR... und *select-ausdruck*).

FETCH Cursor positionieren, Variablen mit Werten aus Satzelementen versorgen, Satz am Bildschirm ausgeben

FETCH positioniert den Cursor auf den nächsten Satz in der Cursortabelle und macht diesen zum aktuellen Satz. Im Programm-Modus werden die Werte der Satzelemente dieses aktuellen Satzes in die angegebene(n) Variable(n) übertragen übertragen. Im Dialog-Modus werden die Werte der Satzelemente dieses aktuellen Satzes am Datensichtgerät angezeigt. Der aktuelle Satz kann von nachfolgenden Anweisungen UPDATE... WHERE CURRENT OF... oder DELETE... WHERE CURRENT OF... geändert oder gelöscht werden, falls der Cursor änderbar ist. Die erste FETCH-Anweisung nach dem OPEN positioniert den Cursor auf den ersten Satz.

```
FETCH cursor [INTO variable,...]
```

cursor	Name des Cursors.
INTO	Die Werte der Satzelemente des aktuellen Satzes werden in die Variablen übertragen. Im Dialog-Modus ist INTO nicht erlaubt. Im Programm-Modus muß INTO angegeben werden.
variable	Name einer Variablen (siehe Handbuch "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Metavariable *variable*). Der ersten Variablen wird der Wert des ersten Satzelements zugewiesen, der zweiten Variablen der Wert des zweiten Satzelements usw. (*variable* entspricht in der SQL-Sprachbeschreibung dem Begriff *benutzervariable*.)

Regeln

- Der Cursor muß vorher mit DECLARE deklariert und mit OPEN geöffnet sein.
- Die Anzahl der Variablen muß gleich sein der Anzahl der Satzelemente in *cursorspezifikation* der Cursor-Deklaration. Außerdem müssen die Datentypen der Variablen und der entsprechenden Satzelemente verträglich sein.

INSERT Sätze in Tabelle einfügen

Bei Angabe einer Liste von Werten nimmt INSERT einen Satz in eine bestehende Basistabelle auf, bei *select-ausdruck* mehrere.
Diese Anweisung können Sie auch für einen änderbaren View angeben.

fett

```
INSERT INTO tabelle

                      ⎧          ⎡wert⎤      ⎫
[(satzelement,...)] ⎨VALUES (⎨NULL⎬,...)⎬ [RETURN INTO variable]
                      ⎩          ⎣*   ⎦      ⎭
                      ⎩select-ausdruck        ⎭
```

---

tabelle Name der Basistabelle oder des änderbaren Views (siehe *tabelle*).

(satzelement,...)
Namen der Satzelemente, in die Werte eingetragen werden sollen.

Fehlt die Angabe, so werden alle Satzelemente der angegebenen Basistabelle bzw. des Views berücksichtigt. Die Reihenfolge der Satzelemente entspricht bei einer Basistabelle der Reihenfolge gemäß der Definition dieser Basistabelle. Bei einem View haben Sie die Reihenfolge mit der Deklaration des View festgelegt.

Jedes *satzelement* muß in der angegebenen Tabelle enthalten sein und darf nicht qualifiziert sein.
Ein einfaches Satzelement darf nicht mehrfach angegeben werden.

VALUES weist den Satzelementen die angegebenen Werte zu. Die Werte werden entsprechend ihrer Reihenfolge den einzelnen Satzelementen zugewiesen. Die Anzahl der angegebenen Werte muß übereinstimmen mit der Anzahl der angegebenen Satzelemente. Sind keine Satzelemente angegeben, muß die Anzahl der Werte der Anzahl der Satzelemente von *tabelle* entsprechen. Außerdem müssen die Werte mit den Datentypen der entsprechenden Satzelemente verträglich sein.

wert Wert, den das Satzelement erhalten soll (siehe Handbuch "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Metavariable *wert*).

NULL Mit diesem Schlüsselwort weisen Sie dem entsprechenden Satzelement den NULL-Wert zu.

* Bei SESAM/SQL:
Das Compound-Key-Attribut wird als automatisches Zählfeld eingesetzt. Dabei darf "*" höchstens einem Satzelement zugewiesen werden. "*" darf nur angegeben werden, wenn das entsprechende Satzelement numerisch und Teil des Compound-Keys ist.

Bei UDS/SQL:
Der Wert für den Primärschlüssel wird vom Datenbanksystem vergeben. "*" muß genau dann angegeben werden, wenn das entsprechende Satzelement der Primärschlüssel ist.

RETURN INTO Je nach eingesetztem Datenbanksystem bewirkt die RETURN INTO-Klausel:

Bei SESAM/SQL:
Der Inhalt des Zählfeldes bei einem Compound-Key wird in die Variable übertragen. Dabei dürfen keine signifikanten Stellen verloren gehen. RETURN INTO ist nur erlaubt, wenn in der VALUES-Klausel "*" enthalten ist.

Bei UDS/SQL:
Der Primärschlüsselwert des aufgenommenen Satzes wird in die Variable übertragen. Dabei dürfen keine signifikanten Stellen verloren gehen. RETURN INTO ist nur erlaubt, wenn für die angegebene Tabelle ein Primärschlüssel (PRIMARY KEY) existiert.

variable Name einer Variablen (siehe Handbuch "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Metavariable *variable*). *variable* muß numerisch deklariert sein.

select-ausdruck
SELECT-Ausdruck, dessen Ergebnistabelle die einzufügenden Sätze liefert. Die Anzahl der Satzelemente in der Ergebnistabelle von *select-ausdruck* muß übereinstimmen mit der Anzahl der Satzelemente der INSERT-Anweisung. Die aus der Ergebnistabelle von *select-ausdruck* übernommenen Werte müssen mit den Datentypen der entsprechenden Satzelemente verträglich sein.
Sind für die INSERT-Anweisung keine Satzelemente angegeben, so muß die Ergebnistabelle von *select-ausdruck* genau soviele Satzelemente enthalten, wie *tabelle*.

Regel

tabelle darf nicht in der FROM-Klausel von *select-ausdruck* oder in einer Unterabfrage innerhalb von *select-ausdruck* vorkommen (siehe *unterabfrage* und *select-ausdruck*).

OPEN Cursor öffnen

OPEN öffnet einen Cursor.

OPEN wertet die Cursor-Spezifikation aus, die Sie bei der Vereinbarung des Cursors mit DECLARE angegeben haben. Dabei verwendet OPEN die aktuellen Werte der Variablen, die in der Cursor-Spezifikation vorkommen und überprüft die Zugriffsrechte.

Durch OPEN wird der bei DECLARE definierte *select-ausdruck* mit den zur Zeit gültigen Werten der in der Cursordeklaration verwendeten Variablen aktualisiert. Anschließend wird der Cursor vor den ersten Satz der Cursortabelle positioniert.

Der Gültigkeitsbereich eines Cursors wird aufgehoben bei
- Programmende
- Programmabbruch
- DRIVE-Ende (STOP).

Gültigkeitsbereich eines Cursors:
Die DRIVE-Anweisungen DROP CURSOR *cursor* und DROP CURSORS sind nur im Dialog-Modus von DRIVE/WINDOWS oder als dynamische Anweisung erlaubt und löschen Cursor.
Beim Übergang vom Dialog- in den Programm-Modus von DRIVE/WINDOWS bleibt die Cursor-Definition erhalten, die Cursor-Position hingegen nicht.
Im Programm-Modus von DRIVE/WINDOWS bleibt ein Cursor, der mit PERMANENT definiert wurde, über das Ende eines mit CALL gerufenen Programms hinaus mit seiner Cursor-Position erhalten.
Der Gültigkeitsbereich des Cursors im Programm-Modus wird aufgehoben beim Übergang in den Dialog-Modus, bei Programmabbruch und DRIVE-Ende (COMMIT WORK WITH STOP).
Ein Cursor, der mit TEMPORARY definiert wurde, wird beim nächsten COMMIT WORK auf höherer Programmebene gelöscht.

```
OPEN cursor
```

cursor Name des Cursors

Regeln

- Der Cursor muß vorher mit DECLARE deklariert sein.
- Der Cursor muß geschlossen sein.

PERMIT Benutzeridentifikation angeben

PERMIT gibt die Benutzeridentifikation für ein verwendetes relationales Schema einer SESAM- bzw. UDS-Datenbank an. Mit der Zuordnung der paßwortgeschützten Datenbank werden der Zugriff auf die Relationen dieser Datenbank erlaubt und der Gültigkeitsbereich aller Bezeichner eröffnet, die in der Datenbank definiert sind. Werden die Benutzeridentifikation und Zuordnung einer paßwortgeschützten Datenbank während des Ablaufs eines DRIVE-Programms benötigt, so muß die PERMIT-Anweisung vor dem Aufruf des Programms angegeben werden.

Maskengesteuerte Angabe

Im UTM-Vorlauf (nach Angabe des TAC) und wenn nicht PERMIT OFF angegeben ist, wird automatisch eine PERMIT-Bildschirm-Eingabemaske ausgegeben, deren Eingabefelder mit Leerzeichen gefüllt sind.
Wird diese Maske ganz oder teilweise ausgefüllt, wird intern eine PERMIT-Anweisung mit den angegebenen Werten bzw. Standardwerten erzeugt.
Die PERMIT-Maske kann beliebig oft ausgefüllt werden, wenn in der Fußzeile 'R' angegeben ist (Voreinstellung). Wird 'N' angegeben, wird eine PERMIT-Anweisung erzeugt und ausgeführt und anschließend der Bildschirm gelöscht. Wird 'S' angegeben, wird die Maske ohne Erzeugen einer PERMIT-Anweisung beendet.

PERMIT bei SESAM-Datenbanken

Der Name des Schemas bei SESAM entspricht dem Namen der SESAM-Datenbank. Solange Sie für ein Schema keine PERMIT-Anweisung angegeben haben, gilt die Standardvereinbarung

– Paßwort: ␣␣␣

Beim Zugriff auf eine nicht-paßwortgeschützte Datenbank ist keine PERMIT-Anweisung erforderlich.

PERMIT bei UDS-Datenbanken Bei einer UDS-Datenbank können Sie auf das relationale Schema nur dann zugreifen, wenn die angegebene Benutzeridentifikation mit dem UDS/SQL-Dienstprogramm BPRIVACY für das relationale Schema vereinbart wurde.

Solange Sie für ein Schema keine PERMIT-Anweisung angegeben haben, gilt die Standardvereinbarung

– Benutzergruppe: SQLGRP

– Benutzername: SQLUSR

– Paßwort: SQLPWD

```
PERMIT SCHEMA = {schema  } [USERGROUP = benutzergruppe] [USERNAME = benutzername]
               {variable}
                          [PASSWORD = passwort]
```

schema — Name des Schemas bzw. der SESAM-Datenbank.
schema darf nicht mit dem Namen eines im aktuellen Programmlauf verwendeten temporären Views übereinstimmen.

variable — Variable, die den Namen des Schemas bzw. der SESAM-Datenbank enthält.

USERGROUP = benutzergruppe
Die Angabe ist nur erlaubt, wenn die aktuelle DRIVE-Umgebung Anschluß an eine UDS-Datenbank bietet.
Wertzuweisung für eine Anwendergruppe (siehe Handbuch "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Metavariable *wert*). *benutzergruppe* muß den UDS-Konventionen genügen. In Programmen muß *benutzergruppe* als Variable (Datentyp CHAR) angegeben werden.

USERNAME = benutzername
Die Angabe ist nur erlaubt, wenn die aktuelle DRIVE-Umgebung Anschluß an eine UDS-Datenbank bietet.
Wertzuweisung für einen Anwendernamen (siehe Handbuch "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Metavariable *wert*). *benutzername* mus^ den UDS-Konventionen genügen. In Programmen muß *benutzername* als Variable (Datentyp CHAR) angegeben werden.

PASSWORD = passwort
Wertzuweisung für ein Paßwort (siehe Handbuch "DRIVE/WINDOWS: Lexikon der DRIVE-Anweisungen" [3], Metavariable *wert*). *passwort* muß den UDS- bzw. SESAM-Konventionen genügen. In Programmen muß *passwort* als Variable (Datentyp CHAR) angegeben werden.

PERMIT OFF Unterdrücken einer UTM-Eingabemaske

PERMIT OFF ist nur in der UTM-Startprozedur zulässig. Die Anweisung unterdrückt das standardmäßige Ausgeben einer Bildschirm-Eingabemaske für die Benutzeridentifikation (PERMIT) im UTM-Vorlauf.

```
PERMIT OFF
```

ROLLBACK WORK Transaktion zurücksetzen

ROLLBACK WORK beendet eine Transaktion und setzt die Änderungen in der Datenbank zurück, die seit Beginn der Transaktion durchgeführt wurden.

Alle innerhalb der Transaktion geöffneten Cursor werden geschlossen.

Ist innerhalb des dynamischen Programmablaufs vor Programmende eine offene Transaktion noch nicht beendet worden, wird von DRIVE/WINDOWS ein ROLLBACK WORK durchgeführt und das Programm mit einer Fehlermeldung abgebrochen.

Die erste fehlerfreie SQL-Anweisung nach ROLLBACK WORK eröffnet eine neue Transaktion.

```
ROLLBACK WORK [WITH RESET]
```

Bezüglich der Datenbank sind ROLLBACK WORK und ROLLBACK WORK WITH RESET identisch.

WITH RESET WITH RESET ist nur im Programm-Modus erlaubt und bezieht sich nur auf die DRIVE-Steuerung. Das Programm wird auf den Stand des letzten COMMIT WORK zurückgesetzt und mit der Anweisung fortgesetzt, die dem COMMIT WORK folgt. Der Inhalt der DRIVE-Variablen wird zurückgesetzt. Wurde seit Programmstart noch kein COMMIT WORK durchlaufen, wird das Programm abgebrochen.

SELECT Daten abfragen

SELECT wählt einen Satz aus Basistabellen oder Views aus. Dabei können Sie Sätze aus mehreren Tabellen miteinander verbinden (Join).
Im Programm-Modus überträgt SELECT die Werte der Satzelemente in die mit INTO angegebenen Variablen. Im Dialog-Modus gibt SELECT die Werte der Satzelemente auf den Bildschirm aus.
Die Anzahl der ausgewählten Sätze (Treffersätze) der SELECT-Anweisung darf nicht größer als 1 sein. Wollen Sie mehr Sätze lesen, müssen Sie einen Cursor verwenden.

```
SELECT [{ALL     }] select-liste
       [{DISTINCT}]

       [INTO variable,... ]

       FROM tabellenangabe,...

       [ WHERE bedingung]

       [GROUP BY satzelement,...]

       [HAVING bedingung]
```

Die angegebene Reihenfolge der Klauseln muß eingehalten werden.

ALL — Standardwert.
Dieser Operand liefert alle mit der WHERE-Klausel ausgewählten Sätze. Auch doppelte Sätze (Duplikate) werden ausgewählt. Beachten Sie, daß beim Vorliegen von Duplikaten somit in jedem Fall ein Fehler auftritt, da als Treffermenge der SELECT-Anweisung maximal ein Satz erlaubt ist.

DISTINCT — Doppelte Sätze werden entfernt.
DISTINCT darf in einer SELECT-Anweisung nur einmal direkt angegeben werden. Eine Unterabfrage innerhalb der SELECT-Anweisung darf jedoch ebenfalls die Angabe DISTINCT enthalten.

GROUP BY — Mit Hilfe der GROUP BY-Klausel werden Tabellensätze in Gruppen zusammengefaßt. Die Ergebnistabelle enthält dann für jede Gruppe einen Satz. Alle NULL-Werte in einer Spalte werden als gleich betrachtet und bilden daher eine Gruppe.

Die GROUP BY-Klausel ist bei der SELECT-Anweisung nicht sehr sinnvoll, da nur ein Treffer möglich ist.

satzelement Satzelement, das ein Gruppierungsmerkmal angibt.

Einschränkung

Wenn Sie die GROUP BY-Klausel angeben, dürfen in der Auswahl der Satzelemente nur noch diejenigen vorkommen, die bei GROUP BY aufgeführt oder Argument einer Mengenfunktion sind.

HAVING bedingung

Es werden Bedingungen angegeben, um Gruppen auszuwählen. Die Ergebnistabelle enthält den Satz für eine Gruppe, wenn die Gruppe die angegebene Bedingung erfüllt. Im Unterschied zur WHERE-Bedingung dürfen in der HAVING-Bedingung auch Mengenfunktionen angegeben werden.

Die HAVING-Klausel ist bei der SELECT-Anweisung nicht sehr sinnvoll, da nur ein Treffer möglich ist.

Satzelemente auswählen

Mit *select-liste* spezifizieren Sie die Satzelemente der Ergebnistabelle (siehe *select-ausdruck*).

INTO Werte von Satzelementen den Variablen zuweisen

Mit INTO weisen Sie den Variablen die Werte der Satzelemente aus der (einzeiligen) Ergebnistabelle zu (siehe *select-ausdruck*).
INTO gilt nur im Programm-Modus.

FROM Tabellen auswählen

In der FROM-Klausel geben Sie Basistabellen oder Views an, aus denen Daten ausgewählt werden (siehe *select-ausdruck*).

WHERE Sätze auswählen

In der WHERE-Klausel geben Sie Bedingungen an, um Sätze für die Ergebnistabelle auszuwählen (Metavariable *select-ausdruck*).
Die Ergebnistabelle enthält nur Sätze, die die angegebenen Bedingungen erfüllen.

SET TRANSACTION Konsistenzlevel festlegen

SET TRANSACTION legt den Konsistenzlevel für eine Transaktion fest. Je höher der Konsistenzlevel ist, desto geringer ist der Grad der Transaktionsparallelität.
Bei SESAM/SQL können Sie mit SET TRANSACTION außerdem den Status für eine Transaktion vereinbaren.

Im folgenden ist die SET TRANSACTION-Anweisung jeweils gesondert für SESAM/SQL und UDS/SQL beschrieben.

Die SET TRANSACTION-Anweisung für SESAM/SQL:

```
                 ⎡                        ⎡0⎤                    ⎤
                 ⎢                        ⎢1⎥  ⎡READ ONLY ⎤      ⎥
                 ⎢CONSISTENCY LEVEL      ⎨2⎬[⎨           ⎬]      ⎥
                 ⎢                        ⎣3⎦  ⎣READ WRITE⎦      ⎥
SET TRANSACTION ⎨                                                ⎬
                 ⎢                                          ⎡0⎤  ⎥
                 ⎢⎡READ ONLY ⎤                              ⎢1⎥  ⎥
                 ⎢⎨           ⎬ [CONSISTENCY LEVEL]        ⎨2⎬  ⎥
                 ⎣⎣READ WRITE⎦                              ⎣3⎦  ⎦
```

Wird die SET TRANSACTION-Anweisung für eine Transaktion nicht angegeben, so gilt der Konsistenzlevel 3 für diese Transaktion.

READ ONLY legt den Transaktionsstatus so fest, daß innerhalb der Transaktion nur lesende Datenbankzugriffe möglich sind.

READ WRITE legt den Transaktionsstatus so fest, daß innerhalb der Transaktion lesende und schreibende Datenbankzugriffe möglich sind.

Falls weder READ ONLY noch READ WRITE angegeben wird, gilt:

- READ ONLY bei Konsistenzlevel 0 oder 1
- READ WRITE bei Konsistenzlevel 2 oder 3

Generell ist garantiert:

- Eine Transaktion wird entweder komplett durchgeführt oder komplett nicht durchgeführt.
- Keine Änderung geht verloren, d.h. "lost updates" sind ausgeschlossen. Unter "lost updates" versteht man folgende Situation:

Wenn Transaktion 1 einen Satz ändert, der anschließend von Transaktion 2 geändert wird, kann bei Zurücksetzen von Transaktion 1 auch die zweite Änderung verloren gehen.

Standardmäßig wird jeder Satz, auf den innerhalb einer Transaktion zugegriffen wird, gegen den Zugriff anderer Transaktionen gesperrt. Bei Wiedergewinnungsanweisungen unterscheidet man in SESAM/SQL außerdem die Modifikationen des Sperrkonzepts "Lesen ohne zu sperren" und "Ignorieren der Sperre".

Die in der SET TRANSACTION-Anweisung festgelegten Konsistenzlevel sind durch folgendes Verhalten beim Lesen gekennzeichnet:

Konsistenz-level	Lesen ohne zu sperren	Ignorieren der Lesesperre
0	ja	ja
1	nein	ja
2	ja	nein
3	nein	nein

Abhängig vom Konsistenzlevel können folgende Phänomene auftreten, wenn zwei Transaktionen parallel auf einen Satz zugreifen:

- schmutziges Lesen - "dirty read"
  Transaktion-1 liest einen Satz mit "Ignorieren der Sperre" (Konsistenzlevel 0 oder 1), der von Transaktion-2 geändert, aber noch nicht freigegeben wurde. Wird Transaktion-2 zurückgesetzt, dann hat Transaktion-1 den geänderten Satz gelesen, der so nicht existiert.
- Nichtwiederholbares Lesen - "non-repeatable read"
  In einer Transaktion wird ein Satz mit "Lesen ohne zu sperren" (Konsistenzlevel 0 oder 2) gelesen, der anschließend von einer zweiten Transaktion geändert wird. Die zweite Transaktion wird beendet und die Änderung festgeschrieben. Liest die erste Transaktion den Satz erneut, dann ist dieser Satz nicht mehr im ursprünglichen Zustand.
- Phantome - "phantoms"
  In einer Transaktion werden Sätze gelesen, anschließend nimmt eine zweite Transaktion weitere Sätze auf. Wiederholt die erste Transaktion die gleiche Suchfrage, kann die Anzahl der Treffersätze größer sein. Phantome können bei allen Konsistenzleveln auftreten.

Die folgende Tabelle zeigt, welche Phänomene bei den einzelnen Konsistenzleveln auftreten können:

Konsistenzlevel	dirty read	non-repeatable read	phantoms
0	x	x	x
1	x	x 1)	x
2		x	x
3			x

zu 1)
tritt bei Sätzen auf, bei denen vorher "dirty read" auftrat.

Regeln

Die SET TRANSACTION-Anweisung muß statisch die erste von PERMIT verschiedene Anweisung einer Transaktion sein.

Die SET TRANSACTION-Anweisung für UDS/SQL:

```
SET TRANSACTION CONSISTENCY LEVEL { 2 }
                                  { 3 }
```

Wird die SET TRANSACTION-Anweisung für eine Transaktion nicht angegeben, so gilt der Konsistenzlevel 2 für diese Transaktion.

Generell ist garantiert:

- Eine Transaktion wird entweder komplett durchgeführt oder komplett nicht durchgeführt.
- Keine Änderung geht verloren, d.h. "lost updates" sind ausgeschlossen. Unter "lost updates" versteht man folgende Situation:
  Wenn Transaktion 1 einen Satz ändert, der anschließend von Transaktion 2 geändert wird, kann beim Zurücksetzen von Transaktion 1 auch die zweite Änderung verloren gehen.
- In keiner Transaktion werden die Änderungen anderer, noch nicht beendeter Transaktionen sichtbar, d.h. "dirty read" ist nicht möglich.

Abhängig vom Konsistenzlevel können folgende Phänomene auftreten, wenn zwei Transaktionen parallel auf einen Satz zugreifen:

- Nichtwiederholbares Lesen - "non-repeatable read"
  In einer Transaktion wird ein Satz gelesen, der anschließend von einer zweiten Transaktion geändert wird. Die zweite Transaktion wird beendet und die Änderung festgeschrieben. Liest die erste Transaktion den Satz erneut, dann ist dieser Satz nicht mehr im ursprünglichen Zustand.
- Phantome - "phantoms"
  In einer Transaktion werden Sätze gelesen, anschließend nimmt eine zweite Transaktion weitere Sätze auf. Wiederholt die erste Transaktion die gleiche Suchfrage, kann die Anzahl der Treffersätze größer sein. Phantome können bei allen Konsistenzleveln auftreten.

Die folgende Tabelle zeigt, welche Phänomene bei den einzelnen Konsistenzleveln auftreten können:

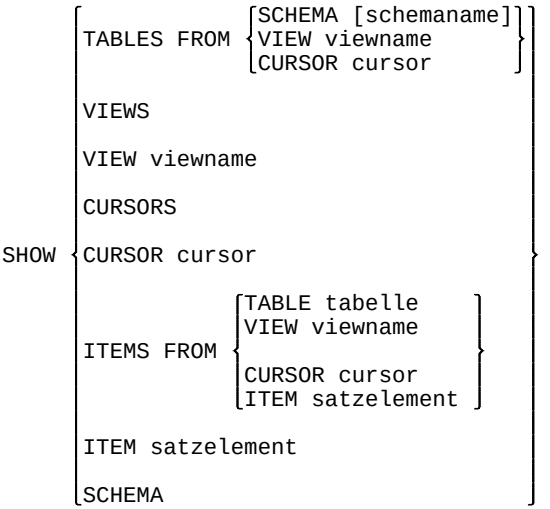
Konsistenzlevel	non-repeatable read	phantoms
2	x	x
3		x

Regel

Die SET TRANSACTION-Anweisung muß statisch die erste von PERMIT verschiedene Anweisung einer Transaktion sein.

SHOW Ausgeben von Informationen über Metadaten

Mit SHOW kann man sich am Bildschirm Informationen über Basistabellen, Views, Cursor, Satzelemente und Schemanamen ausgeben lassen.



TABLES FROM

Für alle Basistabellen des angegebenen Schemas, Views oder Cursors wird der Name der Tabelle ausgegeben.

Wird *schemaname* nicht angegeben, wird *schemaname* von PARAMETER DYNAMIC genommen.

VIEWS Für alle Views wird der Name des Views und "änderbar/nicht änderbar" ausgegeben.

VIEWS darf nur bei UDS-Datenbanken angegeben werden.

VIEW viewname

Für *viewname* wird der Name des Views und "änderbar/nicht änderbar" ausgegeben.

CURSORS Für alle Cursor wird der Name des Cursors und "änderbar/ nicht änderbar" ausgegeben.

CURSORS darf nur bei UDS-Datenbanken angegeben werden.

CURSOR cursor

Für *cursorname* wird der Name des Cursors und "änderbar/ nicht änderbar" ausgegeben.

ITEMS FROM

Für alle Satzelemente der angegebenen Tabelle, des Views oder Cursors wird der Name, der Datentyp mit Stellenanzahl und Nachkommastellen ausgegeben; zusätzlich:

- der Wiederholungsfaktor bei Vektoren oder Wiederholungsgruppen
- der Standardwert, falls vorhanden (bei SESAM-Datenbanken)
- die Kennzeichnung eines Schlüssels
- die Nullwertbedingung
- der Name der referenzierten Tabelle (UDS-Basistabelle)

ITEM satzelement

Für *satzelement* wird dieselbe Information wie bei ITEMS FROM ausgegeben, jedoch nur für das angegebene Satzelement.

satzelement muß mit *prefix* und bei Tabellen mit *schemaname* angegeben werden (siehe Metavariable *satzelement*)

SCHEMA

Bei UDS-Datenbanken werden alle Schemanamen ausgegeben, für die zu diesem Zeitpunkt Zugriffsrechte vereinbart sind.

SCHEMA darf nur bei UDS-Datenbanken angegeben werden.

UPDATE... WHERE bedingung
Werte von Satzelementen in ausgewählten Sätzen ändern

UPDATE... WHERE *bedingung* ändert Werte von Satzelementen in ausgewählten Sätzen einer Basistabelle. Diese Anweisung können Sie auch für einen änderbaren View angeben.

```
UPDATE tabelle SET {satzelement=⎧sqlausdruck⎫
                               ⎨          ⎬},...[ WHERE bedingung]
                               ⎩NULL      ⎭
```

tabelle — Name der Basistabelle oder des änderbaren View (siehe *tabelle*).

SET — weist den Satzelementen die neuen Werte zu, die mit *sqlausdruck* oder dem Schlüsselwort NULL angegeben werden.

Der Datentyp des Ausdrucks und der Datentyp des Satzelements müssen verträglich sein.

satzelement — Das Satzelement muß in der angegebenen Tabelle enthalten sein. *satzelement* darf nicht mit *tabelle* qualifiziert werden.

Ein Satzelement von einfachem Datentyp darf nicht mehrfach angegeben werden. Bei einem Satzelement vom Datentyp Vektor (bei SESAM/SQL: multiples Attribut) dürfen mehrere Ausprägungen angegeben werden. Die angesprochenen Ausprägungen müssen dann paarweise verschieden sein und lückenlos aufeinanderfolgen. Ausprägungsbereiche dürfen nicht angegeben werden.

Bei SESAM/SQL darf ein Primärschlüsselfeld (d.h. der gesamte Schlüssel oder ein Compound-Key-Feld) nicht als Satzelement angegeben werden. Auch bei UDS/SQL darf ein Satzelement nicht den Primärschlüssel bezeichnen.

Die Werte der nicht angegebenen Satzelemente bleiben unverändert.

sqlausdruck — bezeichnet einen Ausdruck (siehe *sqlausdruck*), dessen Wert dem entsprechenden Satzelement zugeordnet wird.
Kommt in *sqlausdruck* ein Satzelement von *tabelle* vor, so gelten die Werte dieses Satzelements vor einer möglichen Änderung mit UPDATE.

NULL — Mit diesem Schlüsselwort weisen Sie dem Satzelement den NULL-Wert zu.

bedingung wählt die Sätze aus, die geändert werden sollen (siehe *bedingung*).

Für jeden Satz der angegebenen Basistabelle oder des angegebenen View wird geprüft, ob *bedingung* erfüllt ist.
Jede Unterabfrage (siehe *unterabfrage*) innerhalb von *bedingung* wird ausgewertet und das Ergebnis beim Abprüfen von *bedingung* verwendet. Falls in der WHERE- oder HAVING-Klausel einer Unterabfrage Satzelemente der zu ändernden Tabelle angesprochen werden, erfolgt die Auswertung der Unterabfrage gesondert für jeden Satz der zu ändernden Tabelle mit den jeweils aktuellen Werten der betroffenen Satzelemente. In der FROM-Klausel einer Unterabfrage innerhalb von *bedingung* darf die zu ändernde Tabelle nicht angesprochen werden.

Fehlt die WHERE-Klausel, so werden alle Sätze der angegebenen Basistabelle oder des angegebenen Views geändert.

Regeln

- *sqlausdruck* darf keine Mengenfunktion (AVG, MIN, MAX, SUM) enthalten.
- Falls *bedingung* eine Unterabfrage enthält, darf *tabelle* nicht in der FROM-Klausel dieser Unterabfrage vorkommen.

UPDATE... WHERE CURRENT OF... Werte von Satzelementen im aktuellen Satz des Cursors ändern

UPDATE... WHERE CURRENT OF... ändert Werte von Satzelementen des Satzes, auf den der Cursor positioniert ist. Diese Anweisung können Sie auch für einen änderbaren View angeben.

Die Position des Cursors bleibt unverändert.

```
UPDATE tabelle SET {satzelement=⎧sqlausdruck⎫
                               ⎩NULL       ⎭},... WHERE CURRENT OF cursor
```

---

tabelle — Name einer Basistabelle oder eines änderbaren Views (siehe *tabelle*). Der angegebene Name der Tabelle muß übereinstimmen mit dem Tabellennamen aus der FROM-Klausel der zugrundeliegenden Cursor-Deklaration.

SET — weist den Satzelementen die neuen Werte zu, die Sie mit *sqlausdruck* oder NULL angeben.

Der Datentyp des Ausdrucks und der Datentyp des entsprechenden Satzelements müssen verträglich sein.

satzelement — gibt das zu ändernde Satzelement an (siehe *satzelement*). Das Satzelement muß in der angegebenen Tabelle enthalten sein.
satzelement darf nicht mit *tabelle* qualifiziert werden.

Ein Satzelement von einfachem Datentyp darf nicht mehrfach angegeben werden. Bei einem Satzelement vom Datentyp Vektor (bei SESAM/SQL: multiples Attribut) dürfen mehrere Ausprägungen angegeben werden. Die angesprochenen Ausprägungen müssen dann paarweise verschieden sein und lückenlos aufeinanderfolgen. Ausprägungsbereiche dürfen nicht angegeben werden.

Die Werte der nicht angegebenen Satzelemente bleiben unverändert.

sqlausdruck — bezeichnet einen Ausdruck (siehe *sqlausdruck*), dessen Wert dem entsprechenden Satzelement zugeordnet wird.
Kommt in *sqlausdruck* ein Satzelement von *tabelle* vor, so gelten die Werte dieses Satzelements vor einer möglichen Änderung mit UPDATE.

NULL Mit diesem Schlüsselwort weisen Sie dem Satzelement den NULL-Wert zu.

cursor Name eines Cursors. Der Cursor muß änderbar sein (siehe DECLARE... CURSOR FOR... und *select-ausdruck*).

Regeln

- Der Cursor muß vorher mit DECLARE deklariert sein.
- Ein *sqlausdruck* darf keine *mengenfunktion* enthalten.
- In der FROM-Klausel der zugehörigen Cursor-Deklaration darf nur ein Tabellenname angegeben sein.
- Der Cursor muß mit der OPEN-Anweisung geöffnet und mit FETCH auf einen Satz positioniert sein.

bedingung Bedingungen vereinbaren

Eine Bedingung besteht aus einem oder mehreren logischen Ausdrücken und den logischen Operatoren AND, OR oder NOT.
Erfüllt ein Satz die vereinbarte Bedingung, wird der Satz in die Ergebnistabelle übernommen.

Eine für relationale Datenbanken charakteristische Bedingung beim Auswählen von Sätzen ist die Join-Bedingung. Die Join-Bedingung ist ein Vergleich von Werten von Satzelementen und dient dazu, aus dem Kartesischen Produkt mehrerer Tabellen nur Sätze auszuwählen, bei denen die Satzelemente die Join-Bedingung erfüllen.

Folgende Auswahlbedingungen können gestellt werden:

- Vergleichen von Ausdrücken durch Vergleichsoperatoren
- Vergleichen eines Ausdrucks mit dem Ergebnis einer Unterabfrage
- Vergleichen eines Ausdrucks mit einem Wertebereich
- Vergleichen eines Ausdrucks mit einer Liste von Werten
- Prüfen, ob ein Satzelement den NULL-Wert enthält
- Vergleichen eines Satzelements mit einem teilweise unbekannten Wert
- Existenzabfrage

NULL-Werte in Bedingungen

Wenn NULL-Werte in Bedingungen vorkommen, kann das Ergebnis von Bedingungen neben erfüllt und nicht erfüllt auch unbestimmt sein. Wann das Ergebnis einer Bedingung erfüllt, nicht erfüllt oder unbestimmt ist, ist jeweils bei der Bedingung beschrieben.
Falls das Ergebnis von WHERE *bedingung* unbestimmt ist, dann reagiert das Datenbanksystem so, als ob die Bedingung nicht erfüllt wäre. Der aktuelle Satz wird dann nicht in die Ergebnistabelle übernommen.

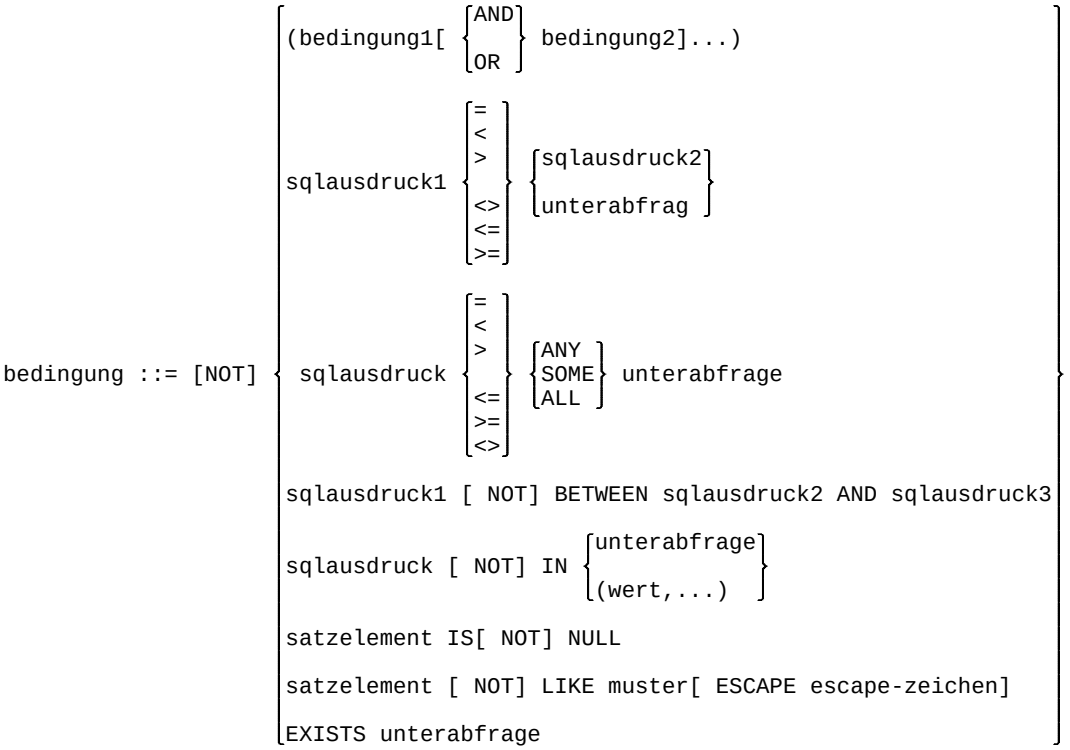
Vergleich alphanumerischer Werte

Zwei Zeichenketten werden von links nach rechts verglichen. Bei unterschiedlich langen Zeichenketten wird die kürzere mit Leerzeichen aufgefüllt. Zwei Zeichenketten sind gleich, wenn sie an jeder Position das gleiche Zeichen haben. Andernfalls legt das erste unterschiedliche Zeichen fest, welche Zeichenkette größer oder kleiner ist.

Vergleich numerischer Werte

Zwei numerische Werte sind gleich, wenn sie dasselbe Vorzeichen und denselben Betrag haben.

Es gilt folgende Syntax für *bedingung*:



Die folgenden Ergebnisse von *bedingung* sind für AND, OR und NOT möglich:

AND
: logisches UND: beide mit AND verknüpften Bedingungen müssen erfüllt sein, damit die gesamte Bedingung erfüllt ist.

AND	*bedingung2*		
bedingung1	erfüllt	nicht erfüllt	unbestimmt
erfüllt	erfüllt	nicht erfüllt	unbestimmt
nicht erfüllt	nicht erfüllt	nicht erfüllt	nicht erfüllt
unbestimmt	unbestimmt	nicht erfüllt	unbestimmt

OR

logisches ODER: mindestens eine der beiden mit OR verknüpften Bedingungen muß erfüllt sein, damit die gesamte Bedingung erfüllt ist.

OR	*bedingung2*		
bedingung1	erfüllt	nicht erfüllt	unbestimmt
erfüllt	erfüllt	erfüllt	erfüllt
nicht erfüllt	erfüllt	nicht erfüllt	unbestimmt
unbestimmt	erfüllt	unbestimmt	unbestimmt

NOT

Negation: die mit NOT verknüpfte Bedingung darf nicht erfüllt sein, damit die gesamte Bedingung erfüllt ist.
Ist das Ergebnis der mit NOT verknüpften Bedingung unbestimmt, so ist auch das Ergebnis der gesamten Bedingung unbestimmt.

bedingung	NOT *bedingung*
erfüllt	nicht erfüllt
nicht erfüllt	erfüllt
unbestimmt	unbestimmt

bedingung Mit *bedingung* werden Sätze aus einer Tabelle ausgewählt.

Regel

Die Datentypen von *sqlausdruck1* und *sqlausdruck2* müssen vergleichbar sein (beide numerisch, alphanumerisch, datetime oder interval).

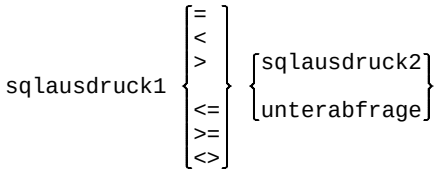
Wenn Sie die logischen Operatoren AND, OR und NOT kombinieren, so gelten die üblichen Vorrangregeln für die Auswertung:

NOT vor AND vor OR

Wenn Sie die beschriebene Reihenfolge ändern wollen, so müssen Sie entsprechend Klammern setzen. Operatoren innerhalb der Klammern haben Vorrang.

Vergleichen von Ausdrücken mit Vergleichsoperatoren

Mit Vergleichsoperatoren können Sie die Werte zweier Ausdrücke bzw. den Wert eines Ausdrucks und das Ergebnis einer Unterabfrage miteinander vergleichen.



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

Die Bedingung ist erfüllt, wenn der Vergleich zutrifft.
Das Ergebnis der Bedingung ist unbestimmt, wenn mindestens ein Ausdruck den NULL-Wert annimmt. Liefert *unterabfrage* kein Ergebnis, so ist das Ergebnis der Bedingung ebenfalls unbestimmt.
Andernfalls ist die Bedingung nicht erfüllt.

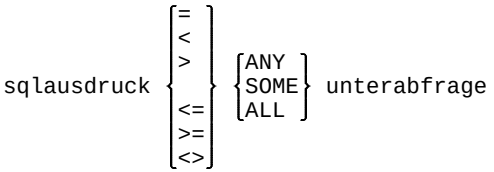
Eine für relationale Datenbanken charakteristische Verwendung des Vergleichs von Werten ist der Join. Bei einem Join wird als Join-Bedingung ein Vergleich angegeben, bei dem beide Operanden Satzelemente sind, deren Werte miteinander verglichen werden.

Regeln

- Die Datentypen von *sqlausdruck1* und *sqlausdruck2* bzw. *sqlausdruck1* und *unterabfrage* müssen vergleichbar sein (beide numerisch, alphanumerisch oder interval).
- Vektoren dürfen nicht in einem Vergleich mit '=', '<>', '<', '>', '<=' oder '>=' vorkommen.
- Die Konstante NULL darf nicht angegeben werden.
- Unterabfrage darf maximal einen Wert als Ergebnis liefern.

Vergleich mit der Ergebnismenge einer Unterabfrage

Mit Vergleichsoperatoren können Sie den Wert eines Ausdrucks mit der Ergebnismenge einer Unterabfrage vergleichen.



Das Ergebnis wird ermittelt, indem der Wert von *sqlausdruck* mit jedem Wert der Ergebnismenge von *unterabfrage* verglichen wird.

ANY, SOME
: Die Bedingung ist erfüllt, wenn der Vergleich für mindestens einen Wert der Ergebnismenge von *unterabfrage* zutrifft.

  Die Bedingung ist nicht erfüllt, wenn *unterabfrage* kein Ergebnis liefert oder der Vergleich für keinen Wert der Ergebnismenge von *unterabfrage* zutrifft.

  Andernfalls ist das Ergebnis der Bedingung unbestimmt.

ALL
: Die Bedingung ist erfüllt, wenn der Vergleich für jeden Wert der Ergebnismenge von *unterabfrage* zutrifft oder wenn *unterabfrage* kein Ergebnis liefert.

  Die Bedingung ist nicht erfüllt, wenn der Vergleich für mindestens einen Wert der Ergebnismenge von *unterabfrage* falsch ergibt.

  Andernfalls ist das Ergebnis des Vergleichs unbestimmt.

Regeln

- Die Datentypen von *sqlausdruck* und der Ergebnismenge von *unterabfrage* müssen verträglich (numerisch oder alphanumerisch) sein.
- Die Konstante NULL darf nicht angegeben werden.
- Vektoren dürfen nicht in einem Vergleich mit "=", "<>", "<", ">", "<=" oder ">=" vorkommen.

Vergleichen eines Ausdrucks mit einem Wertebereich

Es wird geprüft, ob der Wert des Ausdrucks innerhalb oder außerhalb des Wertebereichs liegt.

```
ausdruck1 [NOT] BETWEEN ausdruck2 AND ausdruck3
```

BETWEEN ... AND

Das Ergebnis der Bedingung ist dasselbe wie für die Bedingung *sqlausdruck2* <= *sqlausdruck1* AND *sqlausdruck1* <= *sqlausdruck3*.

Die Bedingung ist erfüllt, wenn der Wert von *sqlausdruck1* innerhalb des Wertebereichs liegt.

NOT BETWEEN ... AND

Das Ergebnis der Bedingung ist dasselbe wie für die Bedingung *sqlausdruck1* < *sqlausdruck2* OR *sqlausdruck1* > *sqlausdruck3*.

Die Bedingung ist erfüllt, wenn der Wert von *sqlausdruck1* außerhalb des Wertebereichs liegt.

Regeln

- Die Datentypen von *sqlausdruck1*, *sqlausdruck2 sqlausdruck3*, müssen verträglich sein (entweder alle alphanumerisch, numerisch, interval oder datumzeit).
- Die Konstante NULL darf nicht angegeben werden.
- Vektoren sind bei einem Vergleich mit BETWEEN... AND nicht zulässig.

Vergleichen eines Ausdrucks mit einer Liste von Werten

Der Ausdruck wird verglichen mit einer Liste von Werten. Diese Werte können Sie explizit angeben oder als Ergebnis einer Unterabfrage bereitstellen.

```
sqlausdruck [NOT] IN {unterabfrage}
                     {(wert,...)  }
```

IN *sqlausdruck* IN *unterabfrage*:
entspricht: *sqlausdruck* = ANY *unterabfrage*

sqlausdruck IN (*wert*,...):
(*wert*,...) wird behandelt analog dem Ergebnis einer Unterabfrage.

NOT IN *sqlausdruck* NOT IN *unterabfrage*:
entspricht: *sqlausdruck* <> ALL *unterabfrage*

sqlausdruck NOT IN (*wert*,...):
(*wert*,...) wird behandelt analog dem Ergebnis einer Unterabfrage.

wert (konst [,konst])
Wert, der alphanumerisch, numerisch oder interval ist und durch eine Konstante oder eine Benutzervariable angegeben wird.

Regeln

- Sie müssen nach IN eine Unterabfrage angeben oder eine Liste von Werten, d.h. mindestens einen Wert.
- Der Datentyp von *sqlausdruck* muß verträglich sein (numerisch oder alphanumerisch)
  - mit dem Datentyp der nach IN spezifizierten Werte bzw.
  - mit dem Datentyp der Werte aus der Ergebnismenge von *unterabfrage*.
- Vektoren dürfen in Vergleichen mit IN nicht vorkommen.

Vergleichen eines Satzelements mit dem NULL-Wert

```
satzelement IS [NOT] NULL
```

satzelement
: Satzelement, das auf den NULL-Wert überprüft werden soll. Der Datentyp des Satzelements kann alphanumerisch oder numerisch sein.

IS NULL
: Die Bedingung ist erfüllt, wenn *satzelement* den NULL-Wert enthält. Andernfalls ist die Bedingung nicht erfüllt.

IS NOT NULL
: Die Bedingung ist erfüllt, wenn *satzelement* nicht den NULL-Wert enthält.
Andernfalls ist die Bedingung nicht erfüllt.

Regel

satzelement darf kein Vektor sein.

Vergleich eines Satzelements mit einem teilweise unbekannten Wert

Der Wert eines Satzelements wird mit einem vorgegebenen Muster (Zeichenkette mit Maskenzeichen) verglichen. erlaubt.

```
satzelement [NOT] LIKE muster [ESCAPE escape-zeichen]
```

satzelement
: Satzelement von alphanumerischem Datentyp.

LIKE
: Die Bedingung ist erfüllt, wenn der Wert von *satzelement* mit *muster* übereinstimmt.

  Das Ergebnis der Bedingung ist unbestimmt, wenn der Wert von *satzlement* der NULL-Wert ist.

  Andernfalls ist die Bedingung nicht erfüllt.

NOT LIKE Die Bedingung ist erfüllt, wenn der Wert von *satzelement* mit *muster* nicht übereinstimmt.

Das Ergebnis der Bedingung ist unbestimmt, wenn *satzelement* den NULL-Wert enthält.

Andernfalls ist die Bedingung nicht erfüllt.

muster Für *muster* geben Sie in ISO/SQL eine alphanumerische Konstante oder eine Variable an.

Das Muster darf aus folgenden Zeichen bestehen:

muster enthält	Die Bedingung ist erfüllt, wenn *satzelement* an derselben Stelle ...
zeichen	genau das Zeichen enthält
_ (Tiefstrich)	ein beliebiges Zeichen enthält
%	eine beliebige Zeichenfolge oder nichts enthält
escape-zeichen%	% enthält
*escape-zeichen*_	_ enthält
escape-zeichen escape-zeichen	*escape-zeichen* enthält

escape-zeichen (zeichen)

vereinbart ein Zeichen, mit dem Sie die Zeichen "%", "_" oder das *escape-zeichen* als Musterzeichen entwerten können. Sie können damit prüfen, ob das Satzelement diese Zeichen enthält.
Für *escape-zeichen* geben Sie eine alphanumerische Konstante oder eine alphanumerische Variable der Länge 1 an.

Regeln

- *satzelement* darf kein Vektor sein.
- *escape-zeichen* muß direkt vor "%", "_" oder einem zweiten *escape-zeichen* stehen.
- *escape-zeichen* und das darauffolgende Zeichen werden in *muster* als ein Zeichen gewertet.
- Die Länge eines Musters, das kein Sonderzeichen "%" und kein *escape-zeichen* enthält, muß übereinstimmen mit der definierten Länge des jeweiligen Satzelements.

Existenzabfrage

Es wird geprüft, ob das Ergebnis der Unterabfrage die leere Menge ist.

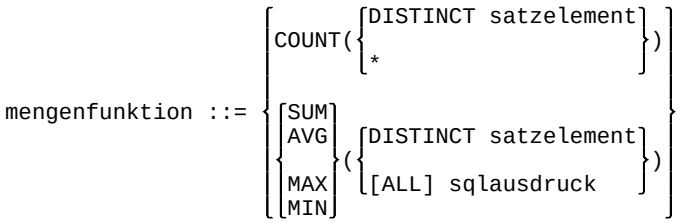
```
EXISTS unterabfrage
```

unterabfrage Unterabfrage, die eine Ergebnistabelle liefert.

Die Bedingung ist erfüllt, wenn die Ergebnistabelle nicht leer ist. Andernfalls ist die Bedingung nicht erfüllt.

mengenfunktion Mengenfunktionen angeben

Eine Mengenfunktion berechnet einen Wert aus einer Menge von Sätzen.



COUNT	zählt die Sätze einer Ergebnistabelle
COUNT (*)	darf nur in *select-liste* eines *select-ausdruck* stehen.
SUM	berechnet die Summe der Werte einer Menge
AVG	berechnet das arithmetische Mittel aus einer Menge von Werten
MAX	bestimmt den größten Wert einer Menge
MIN	bestimmt den kleinsten Wert einer Menge

Das Ergebnis der Mengenfunktion hat folgenden Datentyp:

Mengenfunktion	Datentyp des Ergebnisses
COUNT	Datentyp DECIMAL mit der Genauigkeit 15, keine Nachkommastellen.
MIN und MAX	gleicher Datentyp wie *sqlausdruck* bzw. *satzelement*
SUM	Datentyp DECIMAL mit der Genauigkeit 15. Die Anzahl der Nachkommastellen entspricht der Anzahl der Nachkommastellen des angegebenen *ausdruck*.
AVG	Datentyp DECIMAL mit der Genauigkeit 15. Die Anzahl der Nachkommastellen entspricht 15 minus Anzahl der Stellen vor dem Komma des angegebenen *ausdruck*.

Regeln

- Mengenfunktionen dürfen nur vorkommen
  - in *select-liste* einer SELECT-Abfrage (SELECT-Anweisung, *select-ausdruck*, Unterabfrage).
  - in *sqlausdruck* unmittelbar innerhalb von *bedingung* in einer HAVING-Klausel.
  - in *sqlausdruck* innerhalb von *bedingung* in einer Unterabfrage, wobei die Unterabfrage in einer HAVING-Klausel enthalten sein muß. Das als Argument der Mengenfunktion verwendete Satzelement darf nicht zu einer Tabelle gehören, die in der FROM-Klausel dieser Unterabfrage angegeben wurde. Es muß vielmehr zu einer Tabelle gehören, die in der FROM-Klausel einer übergeordneten SELECT-Abfrage angegeben wurde.
  - Mengenfunktionen dürfen nicht geschachtelt werden. Das heißt, bei einer Mengenfunktion dürfen Sie als Argument keinen sqlausdruck angeben, der selbst wieder eine Mengenfunktion enthält.
  - *satzelement* darf kein Vektor sein. Ebenso darf in *sqlausdruck* kein Vektor vorkommen.

COUNT - Sätze zählen

COUNT zählt die Sätze einer Ergebnistabelle

```
       ⎧DISTINCT satzelement⎫
COUNT (⎨                    ⎬)
       ⎩*                   ⎭
```

DISTINCT	Nur verschiedene Werte des Satzelements werden berücksichtigt. Duplikate und NULL-Werte werden ignoriert.
satzelement	Satzelement mit alphanumerischem oder numerischem Datentyp
*	Alle Sätze der Ergebnistabelle werden berücksichtigt.

Ergebnis

- Bei Argument *satzelement*

  Enthält die Ergebnistabelle nur NULL-Werte, ist das Ergebnis 0.
  Ansonsten werden NULL-Werte ignoriert und das Ergebnis ist wie folgt:

  Ohne GROUP-BY-Klausel in der SELECT-Abfrage:
  Anzahl der verschiedenen Werte von *satzelement*.

  Mit GROUP-BY-Klausel in der SELECT-Abfrage:
  Pro Gruppe in der Ergebnistabelle, die Anzahl verschiedener Werte von Satzelementen in dieser Gruppe.

- Bei Argument *

  Ohne GROUP-BY-Klausel in der SELECT-Abfrage:
  Anzahl der Sätze der Ergebnistabelle. Doppelte Sätze und Sätze, die nur NULL-Werte enthalten, werden mitgezählt.

  Mit GROUP-BY-Klausel in der SELECT-Abfrage:
  Pro Gruppe in der Ergebnistabelle, die Anzahl der Sätze in dieser Gruppe.

SUM - Summe berechnen

SUM berechnet die Summe der Werte einer Menge.

```
     ⎡DISTINCT satzelement⎤
SUM(⎨                    ⎬)
     ⎣[ALL] sqlausdruck   ⎦
```

DISTINCT	Nur verschiedene Werte des Satzelements werden berücksichtigt. Duplikate werden ignoriert.
satzelement	Satzelement mit numerischem Datentyp oder INTERVAL.
ALL	Alle Werte werden berücksichtigt, auch solche die doppelt vorkommen.
sqlausdruck	Ausdruck, der einen numerischen Wert ergibt.

Ergebnis

Die Menge, auf die SUM angewendet wird, wird durch *satzelement* bzw. *sqlausdruck* spezifiziert.

Enthält die Menge nur NULL-Werte, ist das Ergebnis der NULL-Wert.
Ansonsten werden NULL-Werte ignoriert und das Ergebnis ist wie folgt:

Ohne GROUP-BY-Klausel in der SELECT-Abfrage:
Summe der Werte der Menge.

Mit GROUP-BY-Klausel in der SELECT-Abfrage:
Pro Gruppe in der Ergebnistabelle, die Summe der Werte der Menge für diese Gruppe.

Regeln

- *sqlausdruck* muß mindestens ein *satzelement* aus einer Tabelle enthalten, die in der FROM-Klausel einer übergeordneten SELECT-Abfrage angegeben ist.
- *sqlausdruck* darf keine Mengenfunktion enthalten.

AVG - Arithmetisches Mittel

AVG berechnet das arithmetische Mittel aus einer Menge von Werten.

```
     ⎡DISTINCT satzelement⎤
AVG(⎨                    ⎬)
     ⎣[ALL] sqlausdruck   ⎦
```

DISTINCT	Nur verschiedene Werte von *satzelement* werden berücksichtigt. Duplikate werden ignoriert.
satzelement	Satzelement mit numerischem Datentyp.
ALL	Alle Werte werden berücksichtigt, auch solche, die doppelt vorkommen.
sqlausdruck	Ausdruck, der einen numerischen Wert ergibt.

Ergebnis

Die Menge, auf die AVG angewendet wird, wird durch *satzelement* bzw. *sqlausdruck* spezifiziert.

Enthält die Menge nur NULL-Werte, ist das Ergebnis der NULL-Wert.
Ansonsten werden NULL-Werte ignoriert und das Ergebnis ist wie folgt:

Ohne GROUP-BY-Klausel in der SELECT-Abfrage:
Arithmetisches Mittel der Werte der Menge.

Mit GROUP-BY-Klausel in der SELECT-Abfrage:
Pro Gruppe in der Ergebnistabelle, das arithmetische Mittel der Werte für diese Gruppe.

Regeln

- *sqlausdruck* muß mindestens ein *satzelement* aus einer Tabelle enthalten, die in der FROM-Klausel einer übergeordneten SELECT-Abfrage angegeben ist.
- *sqlausdruck* darf keine Mengenfunktion enthalten.

MAX - Maximum bestimmen

MAX bestimmt den größten Wert einer Menge.

```
     ⎡DISTINCT satzelement⎤
MAX(⎨                    ⎬)
     ⎣[ALL] sqlausdruck   ⎦
```

DISTINCT	Die Angabe DISTINCT ist syntaktisch erlaubt, hat aber keine Bedeutung.
satzelement	Satzelement mit numerischem oder alphanumerischem Datentyp.
ALL	Die Angabe ALL ist syntaktisch erlaubt, hat aber keine Bedeutung.
sqlausdruck	Ausdruck, der einen numerischen oder alphanumerischen Wert ergibt.

Ergebnis

Die Menge, auf die MAX angewendet wird, wird durch *satzelement* bzw. *sqlausdruck* spezifiziert.

Enthält die Menge nur NULL-Werte, ist das Ergebnis der NULL-Wert.
Ansonsten werden NULL-Werte ignoriert und das Ergebnis ist wie folgt:

Ohne GROUP-BY-Klausel in der SELECT-Abfrage:
Größter Wert der Menge.

Mit GROUP-BY-Klausel in der SELECT-Abfrage:
Pro Gruppe in der Ergebnistabelle, der größte Wert der Menge für diese Gruppe.

Regeln

- *sqlausdruck* muß mindestens ein *satzelement* aus einer Tabelle enthalten, die in der FROM-Klausel einer übergeordneten SELECT-Abfrage angegeben ist.
- *sqlausdruck* darf keine Mengenfunktion enthalten.

MIN - Minimum bestimmen

MIN bestimmt den kleinsten Wert einer Menge.

```
     ⎡DISTINCT satzelement⎤
MIN(⎨                    ⎬)
     ⎣[ALL] sqlausdruck   ⎦
```

DISTINCT	Die Angabe DISTINCT ist syntaktisch erlaubt, hat aber keine Bedeutung.
satzelement	Satzelement mit numerischem oder alphanumerischem Datentyp.
ALL	Die Angabe ALL ist syntaktisch erlaubt, hat aber keine Bedeutung.
sqlausdruck	Ausdruck, der einen numerischen oder alphanumerischen Wert ergibt.

Ergebnis

Die Menge, auf die MIN angewendet wird, wird durch *satzelement* bzw. *sqlausdruck* spezifiziert.

Enthält die Menge nur NULL-Werte, ist das Ergebnis der NULL-Wert.
Ansonsten werden NULL-Werte ignoriert und das Ergebnis ist wie folgt:

Ohne GROUP-BY-Klausel in der SELECT-Abfrage:
Kleinster Wert der Menge.

Mit GROUP-BY-Klausel in der SELECT-Abfrage:
Pro Gruppe in der Ergebnistabelle, der kleinste Wert der Menge für diese Gruppe.

Regel

- *sqlausdruck* muß mindestens ein *satzelement* aus einer Tabelle enthalten, die in der FROM-Klausel einer übergeordneten SELECT-Abfrage angegeben ist.
- *sqlausdruck* darf keine Mengenfunktion enthalten.

satzelement Satzelemente angeben

```
                  ⎡tabelle⎤  ⎡einf-satzelement         ⎤
satzelement ::= [⎨       ⎬.]⎨                         ⎬
                  ⎣synonym⎦  ⎣vektor(index1[..index2])⎦
```

tabelle
: Name einer Basistabelle oder eines Views (siehe *tabelle*). *tabelle* müssen Sie zur eindeutigen Identifikation angeben, wenn in einer Anweisung identische Namen für Satzelemente aus verschiedenen Tabellen vorkommen.

synonym
: Synonym für eine Tabelle.
synonym kann in der FROM-Klausel der SELECT-Anweisung, der Unterabfrage oder von *select-ausdruck* vereinbart werden. *synonym* müssen Sie zur eindeutigen Identifikation angeben, wenn in einer Anweisung identische Namen für Satzelemente aus verschiedenen Tabellen vorkommen und Sie Synonyme dafür vereinbart haben.

einf-satzelement
: Name eines nicht-strukturierten Satzelements

vektor(index1[..index2])
: Name eines Vektors.
Sie können nur ein Element eines Vektors (indizierter Vektor) mit *index1* oder einen Teilvektor mit *index1..index2* angeben. Die angegebenen Elemente müssen im Vektor enthalten sein.
Für *index1* bzw. *index2* gilt:
index1 und *index2* müssen ganzzahlige numerische Konstanten sein.
index1 muß größer sein als 0;
index2 muß größer oder gleich *index1* sein (bei SESAM/SQL) bzw.
index2 muß größer sein als *index1* (bei UDS/SQL).

Regeln

- Wenn *satzelement* innerhalb der Anweisung nicht eindeutig ist, muß der Name der Tabelle oder das Synonym für die Tabelle dem Satzelement vorangestellt werden.
- *tabelle* und *synonym* dürfen nicht angegeben werden für *satzelement* in einer ORDER BY-Klausel.
- Ein Teilvektor mit *index1..index2* für *satzelement* darf nur in *projektion* einer SELECT-Anweisung oder eines SELECT-Ausdrucks angegeben werden.

select-ausdruck
SELECT innerhalb von SQL-Anweisungen angeben

Der *select-ausdruck* wählt Sätze bzw. Satzelemente aus Basistabellen oder Views aus. Dabei können Sie Sätze aus mehreren Tabellen verbinden (Join).

Das Ergebnis ist wieder eine Tabelle, die Ergebnistabelle. Die Ergebnistabelle enthält die über *select-liste* ausgewählten Satzelemente der Sätze, die Sie über die Selektion mit der WHERE-Klausel auswählen.

Der *select-ausdruck* ermöglicht außerdem die Gruppierung von Ergebnissätzen und das Auswählen von Gruppen.

Der *select-ausdruck* und damit die Ergebnistabelle kann änderbar oder nicht änderbar sein. *select-ausdruck* ist änderbar, wenn die folgenden Bedingungen erfüllt sind:

- In *select-liste* sind nur Satzelemente angegeben; einfache Satzelemente werden nur einmal angesprochen. Teilvektoren dürfen sich nicht überlappen.
- DISTINCT ist nicht angegeben.
- In der FROM-Klausel ist nur eine Tabelle angegeben. Die Tabelle muß eine Basistabelle oder ein änderbarer View sein.
- Die WHERE-Klausel enthält keine Unterabfrage.
- Die Klauseln GROUP BY und HAVING sind nicht angegeben.

```
                              ⎧ALL     ⎫
select-ausdruck ::= SELECT [  ⎨        ⎬] select-liste
                              ⎩DISTINCT⎭

                           FROM tabellenangabe,...

                           [ WHERE bedingung]

                           [ GROUP BY satzelement,...]

                           [ HAVING bedingung]
```

ALL	Standardwert. Auch doppelte Sätze werden ausgewählt.
DISTINCT	Doppelte Sätze werden entfernt.

i Bei SESAM können in *select-liste* max. 6 Funktionen bearbeitet werden.

Satzelemente auswählen

Mit der *select-liste* spezifizieren Sie die Satzelemente der Ergebnistabelle.

```
select-liste ::= ⎧*                ⎫
                 ⎩sqlausdruck,...  ⎭
```

*	Die Ergebnistabelle enthält alle Satzelemente der in der FROM-Klausel angegebenen Tabelle(n). Die Reihenfolge der Satzelemente wird bestimmt durch die Reihenfolge der Tabellen in der FROM-Klausel und innerhalb einer Tabelle durch die definierte Reihenfolge.
tabelle.*	Alle Spalten der Tabelle *tabelle* auswählen. Die Reihenfolge und die Namen der Spalten von *tabelle* werden übernommen.
sqlausdruck	Die Ergebnistabelle enthält die mit *sqlausdruck*,... spezifizierten Satzelemente (siehe *sqlausdruck*) in der angegebenen Reihenfolge.

Regeln

- Die Namen von Satzelementen müssen eindeutig sein. Wenn Sie Basistabellen verbinden und diese Basistabellen Satzelemente mit identischen Namen haben, müssen Sie zur eindeutigen Identifizierung die jeweiligen Namen der Basistabellen bzw. deren Synonyme voranstellen.
- Wenn in einer *select-liste* eine Mengenfunktion vorkommt, dürfen in *select-liste* nur Satzelemente vorkommen, die in der GROUP BY-Klausel aufgeführt sind oder Argument einer Mengenfunktion sind.

Tabellen auswählen

In der FROM-Klausel geben Sie Basistabellen oder Views an, aus denen Daten für die Ergebnistabelle ausgewählt werden.

```
FROM tabellenangabe,...

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

tabellenangabe ::= tabelle[[AS] synonym]
```

tabelle
: Name der Basistabelle oder des View, deren Satzelemente im SELECT-Ausdruck verwendet werden (siehe *tabelle*). Mit der WHERE-Klausel werden dann die Sätze aus dieser Tabelle ausgewählt.
Sind zwei oder mehrere Tabellen angegeben, entsteht eine Ergebnistabelle, die alle Satzelemente der *select-liste* enthält und deren Sätze aus allen möglichen Kombinationen der Sätze der einzelnen Tabellen bestehen.

synonym
: Dieselbe Tabelle kann mehrmals in der FROM-Klausel vorkommen. In diesem Fall müssen Sie Synonyme angeben.
synonym vereinbart einen neuen Namen für *tabelle* für die Dauer der SELECT-Abfrage. *synonym* ist dann innerhalb der gesamten SELECT-Abfrage gültig, sofern nicht in einer Unterabfrage dieser SELECT-Abfrage das gleiche *synonym* für eine andere Tabelle vereinbart wird. In diesem Fall gilt innerhalb der Unterabfrage (und nur dort) die neue Vereinbarung. Entsprechendes gilt auch für beliebig geschachtelte Unterabfragen.

Regeln

- Alle Synonyme innerhalb der FROM-Klausel müssen unterschiedlich sein und dürfen nicht mit einem Tabellennamen übereinstimmen, für den kein *synonym* vereinbart wurde.
- *synonym* ist für die Dauer des SELECT gültig.

Sätze auswählen

In der WHERE-Klausel geben Sie Bedingungen an, um Sätze für die Ergebnistabelle auszuwählen. Die Ergebnistabelle enthält nur Sätze, die die angegebenen Bedingungen erfüllen.

```
WHERE bedingung
```

bedingung — Bedingung, die die ausgewählten Sätze erfüllen müssen. Für jeden Satz der durch die FROM-Klausel festgelegten Tabelle wird geprüft, ob *bedingung* erfüllt ist.
Jede Unterabfrage (siehe *unterabfrage*) innerhalb von *bedingung* wird ausgewertet und das Ergebnis beim Abprüfen von *bedingung* verwendet. Falls in der WHERE- oder HAVING-Klausel einer Unterabfrage Satzelemente der durch die übergeordnete SELECT-Abfrage festgelegten Tabelle angesprochen werden, erfolgt die Auswertung der Unterabfrage gesondert für jeden Satz dieser Tabelle mit den jeweils aktuellen Werten der betroffenen Satzelemente.

Bei mehreren Tabellen in der FROM-Klausel können Sie zusätzlich eine Join-Bedingung angeben, um eine sinnvolle Kombination aus dem Kartesischen Produkt auszuwählen.

Regel

Ein Ausdruck, der unmittelbar in *bedingung* vorkommt, darf kein Satzelement enthalten, das durch eine Mengenfunktion ermittelt wurde. Dagegen darf eine Unterabfrage innerhalb von *bedingung* eine Mengenfunktion enthalten.

Gruppen bilden

Mit Hilfe der GROUP BY-Klausel werden Tabellensätze in Gruppen zusammengefaßt. Die Ergebnistabelle enthält dann für jede Gruppe einen Satz.

```
GROUP BY satzelement,...
```

satzelement — Satzelement, das ein Gruppierungsmerkmal angibt. Satzelemente müssen Sie gegebenenfalls qualifizieren; dabei müssen Sie das Synonym verwenden, wenn Sie die Tabelle in der FROM-Klausel umbenannt haben. Die Reihenfolge der Satzelemente hat keine Bedeutung.

Die Sätze, die in allen angegebenen Satzelementen den gleichen Wert haben, bilden eine Gruppe. Ebenso bilden alle Sätze, die in allen angegebenen Satzelementen den NULL-Wert enthalten, eine eigene Gruppe. Wenn Sie Ergebnissätze in Gruppen zusammenfassen, wirken Mengenfunktionen gruppenweise.

Regel

Jedes *satzelement* der GROUP BY-Klausel muß in einer Tabelle vorkommen, die in der FROM-Klausel angegeben ist.

Gruppen auswählen

In der HAVING-Klausel geben Sie Bedingungen an, um Gruppen auszuwählen. Die Ergebnistabelle enthält für jede Gruppe, die die angegebene Bedingung erfüllt, einen Satz.

```
HAVING bedingung
```

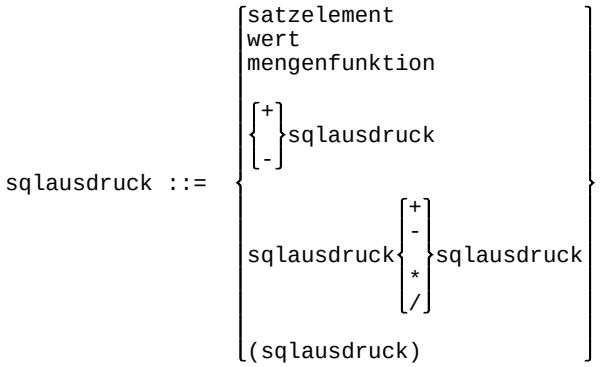
bedingung — Bedingung, die eine Gruppe erfüllen muß. Ist keine GROUP BY-Klausel angegeben, gelten alle Sätze als eine Gruppe.
Im Unterschied zur WHERE-Bedingung dürfen Sie in der HAVING-Bedingung Mengenfunktionen angeben.
Jede Unterabfrage (siehe *unterabfrage*) innerhalb von *bedingung* wird für jede Gruppe ausgewertet und das Ergebnis beim Abprüfen von *bedingung* verwendet. Falls in der WHERE- oder HAVING-Klausel der Unterabfrage Satzelemente der durch die übergeordnete SELECT-Abfrage festgelegten Tabelle angesprochen werden, erfolgt die Auswertung der Unterabfrage gesondert für jeden Satz dieser Tabelle mit den jeweils aktuellen Werten der betroffenen Satzelemente.

Regeln

- Ein Satzelement der mit FROM-, WHERE-, GROUP BY- und HAVING-Klausel spezifizierten Tabelle darf in einer Unterabfrage von *bedingung* nur vorkommen, wenn eine der folgenden Bedingungen gilt:
  - Das Satzelement ist in dieser GROUP BY-Klausel aufgeführt.
  - Das Satzelement ist Argument einer Mengenfunktion.
- Ein Satzelement darf nur dann in *bedingung* unmittelbar angegeben werden, wenn es in der zugehörigen GROUP BY-Klausel angegeben ist oder zu einer Tabelle gehört, die in der FROM-Klausel einer übergeordneten SELECT-Abfrage steht.

sqlausdruck Ausdrücke angeben

Ausdrücke bestehen aus Satzelementen, Literalen, Variablen oder Mengenfunktionen. Ausdrücke können auch zu einem Ausdruck zusammengefaßt werden, indem sie durch arithmetische Operatoren (+,-,*,/) verknüpft werden (Rechenausdruck).



satzelement

bezeichnet den Namen eines Satzelements (siehe Metavariable *satzelement*). Das Satzelement muß in einer Tabelle enthalten sein, die in der FROM-Klausel der SELECT-Abfrage bzw. innerhalb der Anweisungen DELETE, INSERT oder UPDATE angegeben ist.

wert

bezeichnet einen Wert (siehe Metavariable *wert*).

mengenfunktion

bezeichnet eine Mengenfunktion (siehe Metavariable *mengenfunktion*). *sqlausdruck* bezeichnet dann den Wert, den diese Funktion liefert. *mengenfunktion* darf nur in der *select-liste* einer SELECT-Anweisung oder eines *select-ausdruck* vorkommen.

```
[+]
{ }sqlausdruck
[-]
```

"-" bewirkt einen Wechsel des Vorzeichens. "+" läßt den Wert von *sqlausdruck* unverändert. *sqlausdruck* muß numerisch sein. *sqlausdruck* darf nicht mit "+" oder "-" anfangen.
Erlaubt sind damit insbesondere folgende Varianten für numerische Werte bzw. Ausdrücke:

```
     [satzelement   ]
[+]  |wert          |
{-}  |mengenfunktion|
     [(sqlausdruck) ]
```

```
                [+]
                |-|
sqlausdruck     { }sqlausdruck
                |*|
                [/]
```

bezeichnet die Rechenarten Addition, Subtraktion, Multiplikation und Division. Beide Operanden müssen numerisch sein.

(sqlausdruck)

Mit Klammern können Sie Teile von Ausdrücken zu einer Einheit zusammenfassen, um die Reihenfolge der Auswertung von Rechenausdrücken zu verändern. Klammern müssen nach algebraischen Regeln gesetzt werden.

Der Datentyp des ermittelten Ergebnisses eines Rechenausdrucks ist numerisch.
Die Genauigkeit (Anzahl der Vor- und Nachkommastellen) und der Skalenfaktor (Anzahl der Nachkommastellen) des ermittelten Rechenergebnisses sind abhängig

- von der Genauigkeit bzw. vom Skalenfaktor der Rechenoperanden und
- vom verwendeten Rechenoperator.

Wie Sie die jeweilige Genauigkeit und den jeweiligen Skalenfaktor ermitteln können, entnehmen Sie der folgenden Tabelle:

Rechenart	Ergebnis
Addition x+y Subtraktion x-y	Die Genauigkeit P ist gleich der Summe aus dem Maximum der Anzahl von Vorkommastellen und dem um 1 erhöhten Maximum der Anzahl von Nachkommastellen; höchstens aber 15. $P = MIN\ (15,\ MAX(Vor_x,Vor_y) + MAX(S_x,S_y) + 1)$ [1)] Der Skalenfaktor S ist gleich der Anzahl der Nachkommastellen des Operanden, der die maximale Anzahl der Nachkommastellen hat. $S = MAX(S_x,S_y)$ [1)]
Multiplikation x*y	Die Genauigkeit P ist gleich der Summe der Genauigkeiten der Faktoren; höchstens aber 15. $P = MIN\ (15,\ P_x+P_y))$ [1)] Der Skalenfaktor ist gleich der Summe der Skalenfaktoren der Faktoren; höchstens aber 15. $S = MIN(15,\ S_x+S_y)$ [1)]
Division x/y	Die Genauigkeit P ist gleich 15. P = 15 Der Skalenfaktor S errechnet sich nach folgender Formel: $S = MAX(15-P_x+S_x-S_y,\ 0)$ [1)]

[1)] Vor_x - Anzahl der Vorkommastellen des ersten Rechenoperanden
Vor_y - Anzahl der Vorkommastellen des zweiten Rechenoperanden

S_x - Skalenfaktor (Anzahl der Nachkommastellen) des ersten Rechenoperanden
S_y - Skalenfaktor (Anzahl der Nachkommastellen) des zweiten Rechenoperanden

P_x - Genauigkeit des ersten Rechenoperanden
P_y - Genauigkeit des zweiten Rechenoperanden

Es gilt: $P_x = Vor_x + S_x$ $P_y = Vor_y + S_y$

Wenn das mathematische Ergebnis des Rechenausdrucks nach evtl. Abschneiden von Nachkommastellen nicht in das in der obigen Tabelle beschriebene Ergebnisfeld passt, tritt ein Wertüberlauf auf (&SQL_CODE -340). Bei Division durch Null erhält man ebenfalls &SQL_CODE -340. Ist der Skalenfaktor des Rechenergebnisses größer als 15, dann gehen Nachkommastellen verloren.

Regeln

- Wenn in einem Rechenausdruck ein NULL-Wert vorkommt, dann hat der gesamte Ausdruck den Wert NULL.
- Für das Setzen von arithmetischen Operatoren gelten die üblichen algebraischen Regeln.
- Wenn ein *sqlausdruck* eine *mengenfunktion* enthält, muß jedes *satzelement* in der *select-liste* innerhalb einer *mengenfunktion* stehen.
- *satzelemente* von *sqlausdruck* dürfen nicht aus verschiedenen Basistabellen stammen.

tabelle Tabellen angeben

Tabellen, die in SQL-Anweisungen angegeben werden, können Basistabellen oder Views sein. *tabelle* muß zur eindeutigen Identifizierung angegeben werden, wenn in einer Anweisung identische Namen für Satzelemente aus verschiedenen Tabellen vorkommen.

```
                ⎧[schema.]basistabelle⎫
tabelle ::= {   ⎨                     ⎬
                ⎩[schema.]view        ⎭
```

union-ausdruck Ergebnistabellen vereinigen

Der *union-ausdruck* ermöglicht die Vereinigung der Ergebnistabellen zweier SELECT-Ausdrücke.

```
                       ⎧select-ausdruck                                   ⎫
                       ⎪                          ⎧select-ausdruck  ⎫⎪
union-ausdruck ::=     ⎨union-ausdruck UNION[ ALL] ⎨                 ⎬⎬
                       ⎪                          ⎩(union-ausdruck) ⎭⎪
                       ⎩(union-ausdruck)                                  ⎭
```

UNION[ALL]

Die Ergebnistabellen zweier SELECT-Ausdrücke werden vereinigt. Die Ergebnistabelle enthält dann alle Sätze, die in der ersten und zweiten Ergebnistabelle vorkommen. Die Definition von *union-ausdruck* läßt mehrmaliges Anwenden der UNION-Klausel zu. Auf diese Weise können Sie auch mehr als zwei Ergebnistabellen vereinigen.

ALL Doppelte Sätze in der Ergebnistabelle bleiben erhalten.

ALL nicht angegeben:
Doppelte Sätze werden entfernt.

select-ausdruck

siehe *select-ausdruck*

Regel

Bei der Vereinigung mit UNION darf *select-liste* der beteiligten SELECT-Ausdrücke nur Satzelemente enthalten. Außerdem muß *select-liste* für alle mit UNION verbundenen SELECT-Ausdrücke zusammenpassen, d.h. Anzahl und Datentyp der ausgewählten Satzelemente müssen in allen SELECT-Ausdrücken übereinstimmen. Die Angabe "*" bei *select-liste* ist erlaubt.

unterabfrage
Ergebnis eines SELECT als Zwischenergebnis verwenden

Eine Unterabfrage ist eine eingeschränkte SELECT-Abfrage, die eine Ergebnistabelle liefert, die nur aus den Werten für *ein* Satzelement besteht. Unterabfragen können in Bedingungen verwendet werden (siehe *bedingung*).

```
                              ⎡ALL     ⎤  ⎡*           ⎤
unterabfrage ::= (SELECT [{   ⎢        }] {            }
                              ⎣DISTINCT⎦  ⎣sqlausdruck⎦

                            FROM tabellenangabe,...

                            [WHERE bedingung]

                            [GROUP BY satzelement,...]

                            [HAVING bedingung])
```

Die Klauseln von *unterabfrage* sind bei *select-ausdruck* beschrieben. Für *unterabfrage* gilt:

- Wenn die GROUP BY-Klausel nicht angegeben ist und *select-liste* gleichzeitig eine Mengenfunktion enthält, liefert die Unterabfrage als Ergebnis genau einen Wert, nämlich den Wert von *sqlausdruck*. Dabei bildet die durch *select-liste* und die Klauseln spezifizierte (einspaltige) Tabelle das Argument für die in *sqlausdruck* vorkommende(n) Mengenfunktion(en).
- Wenn die GROUP BY-Klausel angegeben ist, wird *sqlausdruck* auf jede Gruppe der (einspaltigen) Ergebnistabelle angewendet. Kommen Mengenfunktionen in *sqlausdruck* vor, so ist jede einzelne Gruppe Argument dieser Mengenfunktionen.
- Wenn die GROUP BY-Klausel nicht angegeben ist und *sqlausdruck* keine Mengenfunktion enthält, wird *sqlausdruck* auf alle Sätze der durch *select-liste* und die Klauseln spezifizierten (einspaltigen) Tabelle angewendet.

Literatur

[1] **DRIVE/WINDOWS** (SINIX)
Software-Produktionsumgebung (SPU)
Benutzerhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Erläuterung der Funktionen der Software-Produktionsumgebung (Arbeitsplatz) und des Expertenmodus. Einsatzvorbereitung für Remote-Zugriff auf BS2000-Datenbanken, für das Erstellen von Anwendungen für das BS2000 und für DRIVE/WINDOWS allgemein.

DRIVE V6.0A (BS2000)
Das Programmiersystem
Benutzerhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
- Einführung in das Programmiersystem DRIVE
- Erläuterung der Funktionen des Dialog-Modus'
- Installationsbeschreibung
- Generierung und Administration von DRIVE im UTM-Betrieb

[2] **DRIVE/WINDOWS** (SINIX)
Programmiersprache
Sprachbeschreibung

Zielgruppe
Anwendungsprogrammierer
Inhalt
Beschreibung der Programmerstellung einschließlich Grafik- und Alpha-Bildschirmformaten sowie Listenformaten mit DRIVE und Report Generator.

DRIVE V6.0A (BS2000)
Beschreibung der Programmiersprache DRIVE

Zielgruppe
Anwendungsprogrammierer
Inhalt
- Erläuterung von SQL-Begriffen und -Konzepten
- Beschreibung der Programmerstellung
- Beispiele

[3] **DRIVE/WINDOWS** (SINIX)
Lexikon der DRIVE-Anweisungen
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-Anweisungen. Meldungen und Schlüsselwörter von DRIVE.

DRIVE V6.0A (BS2000)
Lexikon der Anweisungen
Benutzerhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
- Syntax und Funktionsumfang aller DRIVE-Anweisungen
- Meldungen und Schlüsselwörter von DRIVE

[4] **DRIVE/WINDOWS** (SINIX)
Lexikon der DRIVE-SQL-Anweisungen ffr INFORMIX
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-SQL-Anweisungen für INFORMIX in Kurzform.

[5] **DRIVE/WINDOWS** (SINIX)
Lexikon der DRIVE-SQL-Anweisungen ffr ISO/SQL
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer

Inhalt
Syntax und Funktionsumfang aller DRIVE-SQL-Anweisungen für ISO/SQL in Kurzform.

[6] **DRIVE/WINDOWS** (SINIX)
Lexikon der DRIVE-SQL-Anweisungen für SESAM
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-SQL-Anweisungen für SESAM in Kurzform.

[7] **DRIVE/WINDOWS** (SINIX)
Lexikon der DRIVE-SQL-Anweisungen ffr UDS
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-SQL-Anweisungen für UDS in Kurzform.

[8] **DRIVE/WINDOWS V1.1**
(SINIX)
Ergänzungsband
Benutzerhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Der Ergänzungsband enthält die funktionalen Änderungen von DRIVE/WINDOXS (SINIX) V1.1. Die Handbücher der Version 1.0 werden benötigt.

[9] **DRIVE V6.0A** (BS2000)
Beschreibung der Programmiersprache DRIVE

Zielgruppe
Anwendungsprogrammierer
Inhalt
- Erläuterung von SQL-Begriffen und -Konzepten
- Beschreibung der Programmerstellung
- Beispiele

[10] **DRIVE V6.0A** (BS2000)
Lexikon der Anweisungen
Benutzerhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
- Syntax und Funktionsumfang aller DRIVE-Anweisungen
- Meldungen und Schlüsselwörter von DRIVE

[11] System Interfaces for Applications
SQL für ISO/SQL(BS2000)
Portierbare SQL-Anwendungen für BS2000 und SINIX
Sprachbeschreibung

Zielgruppe
Anwender, die mit SQL oder DRIVE auf SESAM- bzw. UDS-Datenbanken zugreifen wollen.
Inhalt
Das Handbuch beschreibt den Sprachumfang des Produktes ISO/SQL V1.0. Außerdem ermöglicht das Handbuch das Erstellen portierbarer SQL-Anwendungen in BS2000 und SINIX, da der gemeinsame Sprachumfang von ISO/SQL, INFORMIX und SQL-Norm hervorgehoben ist.

[12] **SQL für SESAM/SQL**
Sprachbeschreibung

Zielgruppe
Programmierer, die mit SQL-Anweisungen auf SESAM-Datenbanken zugreifen wollen.
Inhalt
SQL-Anweisungen für den Zugriff auf SESAM-Datenbanken.

[13] **SQL für UDS/SQL**
Sprachbeschreibung

Zielgruppe
Programmierer, die mit SQL-Anweisungen auf UDS-Datenbanken zugreifen wollen.
Inhalt
SQL-Anweisungen für den Zugriff auf UDS-Datenbanken.

[14] **INFORMIX** (SINIX)
SQL
Sprachbeschreibung

Zielgruppe
INFORMIX-Anwender
Inhalt
Vollständige Beschreibung der Datenbanksprache INFORMIX-SQL für alle INFORMIX-Produkte, die eine SQL-Benutzerschnittstelle zur Verfügung stellen. Abweichungen und Erweiterungen gegenüber dem ANSI-Standard sind ebenfalls beschrieben.

[15] **INFORMIX**
Ergänzungsband

Zielgruppe
INFORMIX-Anwender
Inhalt
Der Ergänzungsband für INFORMIX V4.1 enthält die funktionalen Änderungen für die INFORMIX-Produkte: SQL-Sprachbeschreibung, SQL-Nachschlagen, ESQL/COBOL, ESQL/C, C-ISAM und Interaktiver Debugger. Die Handbücher der Version 4.0 werden benötigt.

[16] **Fehlermeldungen für INFORMIX-Produkte** (SINIX)
Benutzerhandbuch

Zielgruppe
INFORMIX-Anwender
Inhalt
Das Handbuch enthält die Texte der Fehlermeldungen, die bei der Arbeit mit INFORMIX-Produkten auftreten können und die dazugehörigen Maßnahmetexte.

[17] **SESAM/SQL** (BS2000)
Aufbau und Wartung
Benutzerhandbuch

Zielgruppe
Datenbank-Verwalter
Inhalt
- Aufbau und Wartung von SESAM-Datenbanken mit dem Datenbank-Administrationsmonitor SESASB
- Schattendatenbankbetrieb

[18] **Dialog Builder V2.0**

[19] **OSF/Motif**
Programmer’s Reference
Release 1.2

Zielgruppe
- Anwendungsentwickler
- Widget-Entwickler

Inhalt
Beschreibung aller Kommandos, Funktionen und Dateiformate des OSF/Motif-Widget-Set.

[20] **SINIX/windows User Environment**
Leitfaden für Experten und Systemverwalter
Benutzerhandbuch

Zielgruppe
Experten in SINIX/windows und Systemverwalter
Inhalt
Das Handbuch erläutert die dem Produkt zugrundeliegenden Konzepte und die Konfiguration der Bedienoberfläche. Die zentralen Clients für Display-Verwaltung, Fensterverwaltung und Verwaltung der Service-Werkzeuge und Dateien werden vorgestellt.

[21] **SINIX/windows User Environment**
Clients zum Nachschlagen
Referenzhandbuch

Zielgruppe
Experten in SINIX/windows und Systemverwalter
Inhalt
Das Handbuch gibt einen vollständigen Überblick über die Clients: über Aufruf, Optionen und Ressourcen, die ihr Aussehen und Verhalten bestimmen. Es erläutert die Reihenfolge und Priorität, in der Ressource-Festlegungen ausgewertet werden.

[22] **X Window System**
Xlib Reference Manual

Zielgruppe
- Anwendungsentwickler
- Widget-Entwickler

Inhalt
Vollständige Beschreibung der C-Programmierschnittstelle der Xlib.

[23] **FORMANT** (SINIX)
Beschreibung

Zielgruppe
- C-Programmierer
- COBOL-Programmierer
- Anwendungsplaner

Inhalt
FORMANT ist eine Maskensteuerung für alle SINIX-Systeme. Das Manual enthält:
- Einführung in FORMANT
- Beschreibung von FORMANTGEN
- Beschreibung der Bedienerschnittstelle
- Programmschnittstellen in C und COBOL
- Beispiele zur Programmierung

[24] **FHS** (TRANSDATA)
Benutzerhandbuch

Zielgruppe
Programmierer
Inhalt
Programmschnittstellen von FHS für TIAM-, DCAM- und UTM-Anwendungen. Erstellen, Einsatz und Verwalten von Formaten.

[25] **IFG für FHS** (TRANSDATA)
Benutzerhandbuch

Zielgruppe
Datenstationsbenutzer, Anwendungsdesigner und Programmierer
Inhalt
Der Interaktive Formatgenerator (IFG) ist ein System zur komfortablen und einfachen Erstellung und Verwaltung von Formaten an Datensichtstationen. Diese Formate können zusammen mit FHS im Verarbeitungsrechner eingesetzt werden. Das Benutzerhandbuch beschreibt, wie die Formate erstellt, geändert und verwaltet werden, sowie die neuen Funktionen von IFG.

[26] **UTM** (SINIX)
Formatierungssystem

Zielgruppe
UTM(SINIX)-Anwender, die mit Formaten arbeiten wollen, C-Programmierer und COBOL-Programmierer
Inhalt
Einsetzen der Formatsteuerung FORMANT in UTM(SINIX)-Teilprogrammen, Erstellen von Formaten, konvertieren von Formaten zwischen BS2000 und SINIX.

[27] **UTM** (SINIX)
Anwendung generieren und administrieren
Benutzerhandbuch

Zielgruppe
Systemverwalter und Administratoren

Inhalt
– Aufbau, Generierung und Betrieb von UTM-Anwendungen auf SINIX
– Arbeiten mit UTM-Meldungen und Fehlercodes.
Einsatz
SINIX-Transaktionsbetrieb

[28] **UTM** (SINIX)
Planen und Entwerfen
Benutzerhandbuch

Zielgruppe
– Organisatoren
– Einsatzplaner
– Programmierer
Inhalt
– Einführung in UTM (SINIX), Erläuterung des Programm-Speicher- und Schnittstellenkonzeptes sowie der Behandlung von Daten, Dateien und Datenbanken
– Hinweise zu Design, Optimierung und Performance von UTM-Anwendungen auf SINIX sowie Datenschutz.
Einsatz
SINIX-Transaktionsbetrieb

[29] **UPIC**
Client-Server-Kommunikation mit UTM
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner Programmierer von UPIC-Programmen
Inhalt
UPIC ermöglicht Programm-Programm-Kommunikation zwischen einer UPIC-Anwendung und einer UTM-Anwendung. Dies funktioniert sowohl lokal mit einer UTM(SINIX)-Anwendung im gleichen Rechner als auch remote mit UTM-Anwendungen auf anderen SINIX- oder BS2000-Rechnern.
Mit UPIC können Sie moderne Präsentationssysteme wie Motif, X Window System an UTM(SINIX) und UTM(BS2000) anbinden. Das Handbuch beschreibt, wie Sie eine UTM-Anwendung in C programmieren.

[30] **ERMS** (SINIX)
Kommandoschnittstelle

Zielgruppe
– Anwender
– Datenbankverwalter

Inhalt
– Konzepte des ERMS aus Anwendersicht
– Beschreibung und Erläuterung der Kommandos für den Anwender

[31] **RADAR V1.0**

[33] **TOM-REF** (BS2000)
Data-Dictionary-System
Benutzerhandbuch

Zielgruppe
Software-Entwickler
Inhalt
Das Handbuch beschreibt Bedeutung und Nutzen des Data-Dictionary-Systems TOM-REF in den Projektphasen der Software-Entwicklung. Es erklärt die Funktionen und die Bedienung des TOM-REF im BS2000 sowie den Daten-Export in ein ERMS-DD im SINIX.

[34] **C-DS C** (SINIX)
Referenzhandbuch für Programmierer

Zielgruppe
C-Programmierer, die unter SINIX V5.41 mit C-DS V1.0 arbeiten.
Inhalt
Beschreibung der Kommandos für die Programmentwicklung, der Bibliotheksfunktionen und Systemaufrufe und einiger Header-Dateien und c-spezifischer Dateiformate.

[35] **C-DS C** (SINIX)
Leitfaden und Werkzeuge für die Programmierung mit C
Benutzerhandbuch

Zielgruppe
C-Programmierer, die unter SINIX V5.41 mit C-DS V1.0 arbeiten.
Inhalt
Im Handbuch werden das C-Übersetzungssystem (Präprozessor, Binder, Include-Dateien, Bibliotheken) und Dienstprogramme zur Entwicklung, Verwaltung, Wartung und Generierung von C-Programmen beschrieben.

[36] SINIX V5.41
Installationsanleitung MX300

Zielgruppe
Service-Techniker und Systemverwalter *Inhalt* Ausführliche Anleitung zur Installation des Betriebssystems SINIX V5.41. Beschreibung der Inbetriebnahme eines vorinstallierten MX300, der Vorbereitungsarbeiten zur Installation von SINIX V5.41 sowie der Installation und Deinstallation von Software.

[37] MX500 (SINIX V5.40)
MX300 (SINIX V5.41)
Referenzhandbuch für Systemverwalter

Zielgruppe
Systemverwalter
Inhalt
Beschreibt Kommandos und Anwendungsprogramme zur Systempflege sowie Dateiformate, spezielle Dateien zur Systemverwaltung und gibt Diagnosehinweise.

[38] **Kommandos** (SINIX V5.40)
Teil 1, A - K
Beschreibung

Zielgruppe
SINIX-Shell-Anwender
Inhalt
Beschreibung der SINIX-Kommandos in alphabetischer Reihenfolge

[39] **Kommandos** (SINIX V5.40)
Teil 2, L - Z
Beschreibung

Zielgruppe
SINIX-Shell-Anwender
Inhalt
Beschreibung der SINIX-Kommandos in alphabetischer Reihenfolge

[40] SINIX V5.41
Leitfaden für Benutzer Benutzerhandbuch

Zielgruppe
Benutzer
Inhalt
Beschreibung der wesentlichen Elemente des SINIX-Betriebssystems. Dazu gehört: Einführung in die Benutzung von SINIX, das Dateisystem, die Prozeßverarbeitung, die Shell.

[41] case/4/0 Methodenhandbuch
* microTOOL GmbH Berlin

[42] case/4/0 Referenzhandbuch
* microTOOL GmbH Berlin

[43] case/4/0 Bedienerhandbuch
* microTOOL GmbH Berlin

[44] **Widget Set**
Programmer's Reference

Zielgruppe
– Anwendungsentwickler
– Widget-Entwickler
Inhalt
Beschreibung aller Kommandos, Funktionen und Dateiformate zu den Widgets XmSniBrowser, XmSniFormat, XmSniHelp, XmSniTable und XmSniTree in der Siemens-Nixdorf-Erweiterung vom OSF/Motif-Widget-Set.

[45] **SINIX-SPOOL**
Anwenden, Verwalten, Programmieren
Benutzerhandbuch

Zielgruppe
Benutzer, Verwalter und Programmierer des SINIX-SPOOL's
Inhalt
Beschreibung der Kommandos, Verwaltungsfunktionen und der C-Schnittstelle zum SINIX-SPOOL

[46] **Styleguide**
Richtlinien zur Gestaltung von Benutzeroberflächen
Benutzerhandbuch

Zielgruppe
Entwickler von Anwendungsprogrammen
Inhalt
Der Styleguide gibt Regeln und Empfehlungen für die Entwicklung einheitlicher Benutzeroberflächen. Es werden jeweils Aufbau, Inhalt und Bedienablauf dargestellt.

[47] SINIX V5.40 (MX500)
SINIX V5.41 (MX300)
Leitfaden für Systemverwalter
Beschreibung

Zielgruppe
Systemverwalter
Inhalt
– Einführung in die Systemverwaltung von SINIX-Systemen
– Anleitung zur Konfigurierung und Wartung des SINIX-Systems

[48] **Kommandos** (SINIX V5.40)
Teil 3, Tabellen und Verzeichnisse
Beschreibung

Zielgruppe
SINIX-Shell-Anwender
Inhalt
Tabellen und Verzeichnisse zu den in Teil 1 und 2 beschriebenen Kommandos – Inhaltsverzeichnis
- Kommando-Übersicht
- Reguläre Ausdrücke
- Sonderzeichen der BOURNE-Shell
- Gerätedateien für Datenträger
- Dateien des SPOOL-Systems in SINIX V5.23 und V5.40
- Zeichensatz ISO 646
- Fachwort, Literatur- und Stichwortverzeichnis

[49] **File Transfer mit SINIX**
FT-SINIX (SINIX) V5.0A
FTOS-SINIX (SINIX) V2.0A
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an SINIX-Benutzer, die FT/FTOS-SINIX nutzen wollen, sowie an den SINIX-Systemverwalter.
Inhalt
Das Handbuch beschreibt die Funktionen von FT-SINIX/FTOS-SINIX. FT-SINIX dient zur Übertragung von Dateien und zum Dateimanagement auf Basis der FTNEA-Protokolle. Das Zusatzprodukt FTSOS-SINIX ermöglicht die Funktionen auf Basis des FTAM-Protokolls.

Mit * markierte Titel sind nicht von der Siemens Nixdorf Informationssysteme AG oder der Siemens AG herausgegeben.

Bestellen von Handbüchern

Die aufgeführten Handbücher finden Sie mit ihren Bestellnummern im *Druckschriftenverzeichnis* der Siemens Nixdorf Informationssysteme AG. Neu erschienene Titel finden Sie in den *Druckschriften-Neuerscheinungen*.

Beide Veröffentlichungen erhalten Sie regelmäßig, wenn Sie in den entsprechenden Verteiler aufgenommen sind. Wenden Sie sich bitte hierfür an Ihre zuständige Geschäftsstelle. Dort können Sie auch die Handbücher bestellen.

Stichwörter

Inhalt

DRIVE/WINDOWS V1.1 (BS2000/SINIX)

Lexikon der DRIVE-SQL-Anweisungen für ISO/SQL

Zielgruppe

Anwendungsprogrammierer

Inhalt

Syntax und Funktionsumfang aller DRIVE-SQL-Anweisungen für ISO/SQL in Kurzform.

Ausgabe: Dezember 1993

Datei: DRV_ISO.PDF

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*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

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*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.