

Select Expressions

SELECT *selection table-expression*

Eine *select-expression* gibt eine Ergebnistabelle an. Sie wird bei den folgenden Statements benutzt:

INSERT | SELECT

Dieses Kapitel behandelt folgende Themen:

- selection
- table-expression

selection

$$\left[\begin{array}{l} \text{ALL} \\ \text{DISTINCT} \end{array} \right] \left\{ \begin{array}{l} \{ \textit{scalar-expression} \text{ [[AS] correlation-name] }, \dots \} \\ * \end{array} \right\}$$

In der *selection* geben Sie an, was ausgewählt werden soll.

ALL/DISTINCT

Doppelt vorkommende Reihen werden nicht automatisch aus dem Ergebnis einer *select-expression* entfernt. Wenn Sie dies wünschen, geben Sie das Schlüsselwort `DISTINCT` an.

Die Alternative zu `DISTINCT` ist `ALL`. Wenn Sie nichts angeben, gilt `ALL`.

scalar-expression

Anstelle von oder zusätzlich zu einfachen Spaltennamen können Sie auch allgemeine *scalar-expressions* angeben, die Skalar-Operatoren und Skalar-Funktionen, die berechnete Werte liefern, enthalten. Siehe *Scalar Expressions*.

Beispiel:

```
SELECT NAME, 65 - AGE
FROM SQL-PERSONNEL
...
```

correlation-name

Es besteht die Möglichkeit, einer *scalar-expression* einen *correlation-name* als Alias-Namen für eine Ergebnisspalte zuzuweisen.

Der *correlation-name* braucht nicht eindeutig sein. Wenn für eine Ergebnisspalte kein *correlation-name* angegeben wird, wird der betreffende *column-name* genommen (falls sich die Ergebnisspalte von einem Spaltennamen ableitet; andernfalls hat die Ergebnisspalte keinen Namen). Der Name einer Ergebnisspalte kann beispielsweise als Spaltenname in der `ORDER BY`-Klausel eines `SELECT`-Statements angegeben

werden.

Stern-Notation (*)

Alle Spalten aller in der FROM-Klausel angegebenen Tabellen werden ausgewählt.

Beispiel:

```
SELECT *
FROM SQL-PERSONNEL, SQL-AUTOMOBILES
...
```

table-expression

```
FROM table-reference,...
[WHERE search-condition]
[GROUP BY column-reference,... ]
[HAVING search-condition]
```

Die *table-expression* gibt an, von wo und nach welchen Kriterien Reihen gelesen werden sollen.

table-reference

```
{ table-name [correlation-clause]
  [TABLE] subquery correlation-clause
  joined-table
  TABLE function-name (scalar-expression,...) correlation-clause }
```

In der FROM-Klausel geben Sie eine oder mehrere Tabellen an, die die in der *selection-list* verwendeten Spaltenfelder enthalten müssen.

Es besteht die Möglichkeit, einem *table-name* eine *correlation-clause* zuzuweisen.

Sie können entweder eine einzelne Tabelle angeben oder eine Zwischenergebnistabelle erzeugen, die das Ergebnis einer *subquery* oder einer Join-Operation ist (siehe unten).

Da in einer FROM-Klausel mehrere Tabellen (d.h. DDMs) angesprochen werden können und eine *table-expression* mehrere FROM-Klauseln enthalten kann, wenn *subqueries* angegeben sind, bestimmt die Datenbanknummer (DBID) des ersten DDMs in der ersten FROM-Klausel die Datenbank.

Bei einer *subquery* muss eine *correlation-clause* zugewiesen werden.

Die Klausel TABLE *function-name* gehört zum SQL Extended SET, und es ist eine *correlation-clause* mit einer *column-name*-Liste erforderlich.

Ein `FINAL TABLE`-Schlüsselwort, gefolgt von einem `INSERT`-Statement in Klammern gehört zum SQL Extended Set und gibt an, dass die eingefügten Reihen für das betreffende `SELECT`-Statement zurückgegeben werden. Die Ergebnistabelle beinhaltet alle Reihen, die eingefügt wurden. Alle Spalten der eingefügten Tabelle können in der *select list* referenziert werden. Wenn das `INSERT`-Statement in der *table-reference* benutzt wird, kann das *subselect* noch die `WHERE`-Klausel, `GROUP BY`-Klausel, `HAVING`-Klausel und *aggregate-functions* angeben.

correlation-clause

`[AS] correlation-name [(column-name,...)]`

Eine *correlation-clause* besteht aus `KEYWORD AS` als Option und einem *correlation-name*, und es folgt ihr als Option eine einfache *column-name*-Liste. Die *column-name*-Liste gehört zum SQL Extended Set.

joined-table

<i>table-reference</i>	[{	<code>INNER</code> <code>LEFT [OUTER]</code> <code>RIGHT [OUTER]</code> <code>FULL [OUTER]</code>	}]	<code>JOIN table-reference ON join-condition</code>
------------------------	---	---	--	---	---	---

Eine *joined-table* gibt eine Zwischenergebnistabelle an, die das Ergebnis einer Join-Operation ist.

Der Join kann ein `INNER`, `LEFT OUTER`, `RIGHT OUTER` oder `FULL OUTER JOIN` sein. Falls Sie nichts angeben, gilt `INNER`.

Es ist möglich, mehrere Joins zu schachteln, d.h. die Tabellen, die die Zwischenergebnistabelle bilden, können ihrerseits Zwischenergebnstabellen einer Join-Operation oder einer *subquery* sein, wobei letztere wiederum ebenfalls eine *joined-table* oder eine weitere *subquery* in der `FROM`-Klausel haben kann.

join-condition

Bei `INNER`, `LEFT OUTER` und `RIGHT OUTER` Joins:

`search-condition`

Bei `FULL OUTER` Joins:

`full-join-expression = full-join-expression [AND ...]`

full-join-expression

{	<i>column-name</i>	}
{	<code>VALUE</code> <code>COALESCE</code>	}
}	<i>(column-name , ...)</i>	}

In einer *join-expression* sind nur *column-names* und die *scalar-function* VALUE (bzw. ihr Synonym COALESCE) erlaubt. Siehe *column-name*.

WHERE-Klausel

[WHERE *search-condition*]

In der WHERE-Klausel geben Sie eine Suchbedingung (*search-condition*) an, nach der die Reihen gelesen werden sollen.

Beispiel:

```
DEFINE DATA LOCAL
01 NAME      (A20)
01 AGE       (I2)
END-DEFINE
...
SELECT *
  INTO NAME, AGE
  FROM SQL-PERSONNEL
  WHERE AGE = 32
END-SELECT
...
```

Siehe *search-condition*.

GROUP BY-Klausel

[GROUP BY *column-reference, ...*]

Die GROUP BY-Klausel sortiert die in der FROM-Klausel angegebene Tabelle nach Gruppen, und zwar so, dass alle Reihen einer Gruppe in der GROUP BY-Spalte den gleichen Wert haben. Jede *column-reference* in der Selektionsliste muss entweder eine GROUP BY-Spalte sein oder mit einer *aggregate-function* angegeben werden. *Aggregate-functions* werden auf einzelne Gruppen (nicht auf die ganze Tabelle) angewandt. Die Ergebnistabelle enthält soviele Reihen wie Gruppen.

Siehe *column-reference* und *aggregate-function*.

Beispiel:

```
DEFINE DATA LOCAL
1 #AGE      (I2)
1 #NUMBER   (I2)
END-DEFINE
...
SELECT AGE , COUNT(*)
  INTO #AGE, #NUMBER
  FROM SQL-PERSONNEL
  GROUP BY AGE
...
```

Steht vor der GROUP BY-Klausel eine WHERE-Klausel, werden vor dem Aussortieren nur diejenigen Reihen von der GROUP BY-Klausel erfasst, die die WHERE-Bedingung erfüllen.

HAVING-Klausel

[HAVING *search-condition*]

Wenn Sie eine HAVING-Klausel verwenden, sollten Sie auch eine GROUP BY-Klausel verwenden. Genau wie die WHERE-Klausel Reihen aus einer Ergebnistabelle aussortiert, sortiert die HAVING-Klausel Gruppen aus, und zwar auf Grundlage einer Suchbedingung (*search-condition*). *Scalar-expressions* in einer HAVING-Klausel dürfen pro Gruppe nur einen Wert enthalten.

Siehe *scalar-expression* und *search-condition*.

Beispiel:

```
DEFINE DATA LOCAL
1 #NAME      (A20)
1 #AVGAGE    (I2)
1 #NUMBER    (I2)
END-DEFINE
...
SELECT NAME, AVG(AGE), COUNT(*)
  INTO #NAME, #AVGAGE, #NUMBER
  FROM SQL-PERSONNEL
  GROUP BY NAME
  HAVING COUNT(*) > 1
...
```