SQL-Kurzreferenz

Select-Anweisungen

```
SELECT [DISTINCT] {* | Spalte1
[ [AS ] "Alias" ], ... }
FROM Tabellenname;
```

Arithmetische Operatoren
```
SELECT Spalte1, Spalte2 + Wert
FROM Tabellenname;
```

Alias Definition
```
SELECT Spalte1 [AS] Alias,
FROM Tabellenname  TabAlias;
```

Bedingungen mit WHERE

```
SELECT [DISTINCT] {* | Spalte [ [AS ]
" Alias ], ... } FROM Tabelle
[ WHERE Bedingung(en) ] ;
```

Mögliche Operatoren
```
=; >; >=; <; <=; <>;
IS NULL; IS NOT NULL;
BETWEEN ... AND ... ;
IN (Liste); LIKE
```

Verwendung
```
WHERE Spalte IS NULL
WHERE Spalte BETWEEN ... AND ...
WHERE Spalte IN (Eintr.1, Eintr.2,...)
WHERE Spalte LIKE '[%][_]String[%][_]'
% beliebig viele Zeichen (auch null)
_ ein beliebiges Zeichen
```

Logische Operatoren (AND, OR, NOT)
```
WHERE Bedingung1 AND Bedingung2
WHERE Spaltenname NOT IN (Liste)
Reihenfolge der Wichtigkeit: Klammern;
Vergleichsoperatoren; NOT; AND; OR
```

Sortierung mit ORDER BY
```
SELECT * FROM Tabellenname
[ WHERE Bedingung(en) ]
[ ORDER BY { Spaltenname | Ausdruck |
Aliasname [ ASC | DESC ] } ]
Aufsteigend (asc)(Default)
Absteigend (desc)
```

JOINS

Equijoin
```
SELECT {Alias1.Spalte1,
Alias1.Spalte2, Alias2.Spalte1, ...}
FROM Tab1 Alias1, Tab2 Alias2, ...
WHERE Alias1.Spalte1 = Alias2.Spalte1
[AND Alias2.Spalte2 = Alias3.Spalte1];
```

Alternativ:
```
SELECT {Alias1.Spalte1,
Alias1.Spalte2, Alias2.Spalte1, ...}
FROM (Tab1 Alias1 INNER JOIN Tab2
Alias2 ON Alias1.Spalte1 =
Alias2.Spalte1)
INNER JOIN Tab3 Alias3
ON Alias2.Spalte2 = Alias3.Spalte1
WHERE (...);
```

Outer-Join

```
SELECT {Alias1.Spalte1,Alias1.Spalte2,
Alias2.Spalte1, ...}
FROM (Tab1 Alias1 {LEFT|RIGHT|FULL|
OUTER} JOIN Tab2 Alias2
ON Alias1.Spalte1 = Alias2.Spalte2)
WHERE (...);
```

LEFT JOIN orientiert sich an der Tabelle 1 und ergänzt fehlende Informationen mit NULL-Datensätzen der Tabelle 2.

RIGHT JOIN orientiert sich an der Tabelle 2 und ergänzt fehlende Informationen mit NULL-Datensätzen der Tabelle1.

Self-Join

```
SELECT {Alias1.Spalte1,
Alias1.Spalte2, Alias2.Spalte1, ...}
FROM Tab1 Alias1, Tab1 Alias2
WHERE Alias1.Spalte1 = Alias2.Spalte2;
```

```
alternativ:
SELECT {Alias1.Spalte1,
Alias1.Spalte2, Alias2.Spalte1, ...}
FROM (Tab1 Alias1 INNER JOIN Tab1
Alias2 ON Alias1.Spalte1 =
Alias2.Spalte2) WHERE (...);
```

Gruppenfunktionen

```
SELECT Gruppenfkt.(Spaltenname), ...
FROM Tabelle [WHERE Bedingung(en)]
[ORDER BY {Spaltenname|Ausdruck|
Aliasname} [ASC|DESC] ];

AVG (Spaltenname) : Durchschnitte
SUM (Spaltenname) : Summe
MIN (Spaltenname) : Minimum
MAX (Spaltenname) : Maximum
COUNT (Spaltenname) : Anzahl
```
NULL-Werte werden von den Funktionen nicht berücksichtigt

```
COUNT (*) (Zählt Zeilen mit NULL mit)
```

Datengruppen mit GROUP BY
```
SELECT Spalte1,
Gruppenfunktion(Spalte2), ...
FROM Tabelle
[ WHERE Bedingung(en) ]
[ GROUP BY Spaltenname1 [, ...] ]
[ HAVING Gruppenbedingung ]
[ ORDER BY {Spaltenname1 | Ausdruck |
Aliasname} [ ASC | DESC ] ];

HAVING dient der Einschränkung
Gruppenergebnisse ein.
```

Unterabfragen

SELECT-Unterabfragen
```
SELECT Spalten FROM Tabelle
WHERE Spaltenname Operation
(Select-Statement) [ AND ... ];
```

Select darf nur einen Wert als Vergleichswert zurückliefern. Unterabfragen, die mehrere Werte zurückliefern müssen die Operatoren IN; ANY; ALL; EXISTS verwenden.

Beispiel:
```
SELECT A.A_NR FROM ARTIKEL As A
WHERE EXISTS
  (SELECT B.UMSATZ_NR FROM UMSATZ As B
       WHERE B.A_NR = A.A_NR)
```

Beispiel ALL / ANY:
```
SELECT * FROM Waggons
        WHERE waggon_id < [ALL|ANY]
  (SELECT waggon_id FROM Kunden);
```
Alle ids aus Kunden müssen größer als waggon_id sein. Bei ANY muss Übereinstimmung nicht bei allen Elementen der Ergebnismenge vorliegen.

UPDATE Unterabfragen
```
UPDATE Tabelle Alias SET Spalte =
(SELECT expr FROM Tabelle alias2
WHERE Alias.Spalte = "5"
```

DELETE Unterabfragen
```
DELETE FROM Tab1 Alias1 WHERE Spalte
Operator (SELECT expr FROM Tab)
```

Mengenoperationen

Anzahl und Typ der SELECT-Anweisungen müssen übereinstimmen.

Vereinigung
```
SELECT Spalten FROM Tabelle
[WHERE Bedingung(en)]
UNION SELECT Spalten
FROM Tabelle [WHERE Bedingung(en)]
```

Durchschnitt
```
SELECT Spalten FROM Tabelle
[WHERE Bedingung(en)]
INTERSECT SELECT Spalten FROM Tabelle
[WHERE Bedingung(en)] ;
```

Differenz
```
SELECT Spalten FROM Tabelle
[WHERE Bedingung(en)]
MINUS SELECT Spalten FROM Tabelle
[WHERE Bedingung(en)];
```

Tabelleninhalt bearbeiten

Datensätze einfügen
```
INSERT INTO Tab[(Spalte1,Spalte2,...)]
VALUES (Wert1, "Wert2",...);
```

Datensätze ändern
```
UPDATE Tabelle SET Spalte1 = Wert1,
[Spalte2 = Wert2, ...]
[WHERE Bedingung(en)];
```

Datensätze löschen
```
DELETE FROM Tabelle
[WHERE Bedingung(en)];
```

DDL-Data Definition Language

Datenbank erstellen / löschen
```
CREATE DATABASE datenbankname;
DROP DATABASE datenbankname;
```

Tabelle erstellen
```
CREATE TABLE tabellenname
(spaltenname datentyp [NOT NULL],
 [...,]
 spaltenname datentyp[NOT NULL]);
```

Datentypen: CHAR(n), INT, SMALLINT, NUMBER, FLOAT(n), REAL, DOUBLE PRECISION, DEC(m, [n]), DATE

Tabelle löschen
```
DROP TABLE tabellenname
```

Spalten hinzufügen
```
ALTER TABLE tabellenname
    ADD spalte datentyp [NOT NULL],
    [...,];
```

Spalte löschen
```
ALTER TABLE tabellenname
    DROP (spalte,[... ,], spalte);
```