Leseprobe: **SQL mit MySQL - Band 4 Kompendium mit Online-Übungs-DB**

Kompendium zur schnellen Kurzinformation der Datenbanksprache SQL/MySQL 5.1

im Internet: www.Datenbanken-Programmierung.de

...

3.0 SQL nach thematischen Zusammenhängen

In den nachfolgenden Kapiteln werden SQL-Befehle, -Klauseln und -Funktionen nach thematischen Zusammenhängen geordnet dargestellt, gegebenfalls kurz erläutert und mit Beispielen unterlegt.

3.1 Befehle und Klauseln

Die Abfragen, auch als Anweisungen oder Befehle bezeichnet, bestehen je nach Zielsetzung in der Regel aus mehreren Klauseln: Select, From, Where, Group by, Having oder Order by.

In einer Where-Klausel werden Bedingungen für das Sammeln der Daten mit Hilfe von Vorgabewerten, Operatoren (<, =, >, <>, ...) und Wildcards (_, %) formuliert.

3.2 SELECT

SELECT [ALL | DISTINCT | DISTINCTROW]

```
[HIGH_PRIORITY]   [STRAIGHT_JOIN]
select_expr, ...
FROM  tab_ref
[WHERE condition]
[GROUP BY  { col | expr | col_position }  [ASC | DESC] ]
[HAVING condition]
[ORDER BY { col | expr | col_position } [ASC | DESC] ]
[LIMIT  { [col_no, ] number_of_rows } ]
[PROCEDURE  proc_name( argument_list ) ]
[INTO OUTFILE  'file_name'  export_options
   |  INTO DUMPFILE  'file_name' ]
[FOR UPDATE  |  LOCK IN SHARE MODE]
```

Hinweise:
select_expr: * für alle Spalten oder mindestens eine Spalte: col oder mehrere Spalten: col1, col2, ...
tab_ref: die referenzierten Tabellen, Syntax siehe 4.0 Joins

ALL: Standard-Einstellung - es werden alle Datensätze gelistet. Bei DISTINCT oder DISTINCTROW (Synonym) wird die Ausgabe von Duplikaten unterbunden.
HIGH_PRIORITY: Anweisung wird vorrangig vor allen anderen Anweisungen und auch bei Sperrungen ausgeführt.
STRAIGHT_JOIN: zwingende Vorgabe von Verknüpfungen und Reihenfolgen

Beispiele:

```
SELECT   *    FROM  tab ;
SELECT   col   FROM  tab ;
SELECT   col1, col2, ...  FROM  tab ;
SELECT   tab.col   FROM  tab ;
SELECT   CONCAT( col1, '-', col2 )  FROM tab ;
```

ALIAS

SELECT col AS col_alias FROM tab ;

SELECT col col_alias FROM tab ;

SELECT tab_alias.col FROM tab AS tab_alias ;

SELECT tab_alias.col FROM tab tab_alias ;

SELECT tab_alias.col col_alias FROM tab tab_alias ;

Hinweis: das Schlüsselwort AS kann bei der Vergabe von Alias-Name weggelassen werden.

DISTINCT : keine Duplikate

SELECT DISTINCT col FROM tab ;

SELECT DISTINCT tab1.col FROM tab1, tab2
WHERE tab1.col = tab2.col ;

WHERE mit AND, OR und NOT

SELECT col FROM tab WHERE col operator value ;

SELECT col FROM tab WHERE NOT condition ;

SELECT col FROM tab WHERE condition1 AND condition 2 ;

SELECT col FROM tab WHERE condition1 OR condition2 ;

Hinweis:: condition z.B.: col >= 12 oder colX = ‘value‘ odgl.

BETWEEN

SELECT col FROM tab WHERE col BETWEEN value1 AND value2 ;

IN

SELECT col FROM tab WHERE col IN (value1, value2, ..., value-n) ;

SELECT col FROM tab WHERE col NOT IN (value1, value2, ..., value-n) ;

LIKE

```
SELECT  col  FROM  tab  WHERE  col  LIKE  ‘pattern‘ ;
```

Hinweis pattern:: Wildcards: _ und %

ORDER BY

```
SELECT  col  FROM  tab   ORDER BY  col  DESC ;
SELECT  col1, col2, col3  FROM  tab   ORDER BY  2  DESC ;
SELECT  col1, col2, col3  FROM  tab   ORDER BY  MID(col1, 5, 2)  DESC ;
SELECT  col  FROM  tab   WHERE  colX > 0  ORDER BY  col1, col2 ;
```

GROUP BY

```
SELECT  col1,  SUM(col2)  FROM  tab   GROUP BY  col1 ;
SELECT  col1,  SUM(col2)  FROM  tab   GROUP BY  col1  DESC;
```

HAVING

```
SELECT  count( col )  AS  col_alias  FROM  tab
  GROUP BY  col_alias  HAVING  col_alias = 12 ;
SELECT  col1, MAX( col2 )  FROM  tab
  GROUP BY  col1  HAVING  MAX( col2 ) >= 12 ;
```

Hinweis: Having ist nach SQL-Standard nur in Zusammenhang mit der Group-by-Klausel anwendbar. Bei MySQL ist Having jedoch auch für alle gelisteten Spalten der Select-Anweisung und Spalten der äußeren Unterabfrage einsetzbar.

LIMIT

```
SELECT  *  FROM  tab  LIMIT 7 ;
SELECT  *  FROM  tab  LIMIT 7, 10 ;
```

Hinweis: bei LIMIT 7 werden maximal 7 Ergebniszeilen ausgegeben. Bei Limit 7, 10 werden ab der 7 Zeile die folgenden 10 Ergebniszeilen gelistet.

PROCEDURE

```
SET @var = 7 ;
PREPARE  test1  FROM   'SELECT *  FROM  tab   LIMIT ?' ;
EXECUTE  test1  USING  @var ;
SET @var = 7 ;   SET  @anzahl =10 ;
PREPARE  test2  FROM   'SELECT *  FROM  tab   LIMIT  ?, ? ' ;
EXECUTE  test2  USING  @var, @anzahl ;
```

Hinweis: in vorbereitenden Anweisungen können Variablen eingesetzt werden.

INTO OUTFILE

```
SELECT  col1, col2, col3, col4   INTO OUTFILE  'tmp/test.txt'
  FIELDS TERMINATED BY  ','   OPTIONALLY ENCLOSED BY ' " '
  LINES TERMINATED BY  '\n'
  FROM  tab ;
```

Hinweis: die Ausgabedatei wird auf dem Server abgelegt. Bereits vorhandene Dateien werden nicht überschrieben.

INTO DUMPFILE

Ablage eines einzigen Datensatzes ohne Trennzeichen.

Ergänzungen Select

```
SELECT ...
  [SQL_SMALL_RESULT]  [SQL_BIG_RESULT]  [SQL_BUFFER_RESULT]
  [SQL_CACHE  |  SQL_NO_CACHE]  [SQL_CALC_FOUND_ROWS]
...
```

für Geschwindigkeitsoptimierungen bei großen Datenmengen.

SQL_CALC_FOUND_ROWS: für Ergebnisanzahl unabhängig von LIMIT. Die Anzahl kann mit SELECT FOUND_ROWS() angezeigt werden.

3.3 INSERT

```
INSERT [INTO] tab [ ( col1, col2, ... ) [, ( col3, col4, ... ), ...] ]
  VALUES ( {expr | DEFAULT}, ... )
  [ ON DUPLICATE KEY UPDATE colX = expr, ... ]
oder
INSERT [INTO] tab
  SET ( {expr | DEFAULT}, ... )
  [ ON DUPLICATE KEY UPDATE colX = expr, ... ]
oder
INSERT [INTO] tab [ ( col1, col2, ... ) [, ( col3, col4, ... ), ...] ]
  SELECT ...
  [ ON DUPLICATE KEY UPDATE colX = expr, ... ]
```

Beispiele:

```
INSERT INTO tab (col1, col2, col3, col4)
  VALUES (value1, value2, FLOAT(value3,5,2), value4) ;
INSERT INTO tab  VALUES (value1, value2, value2, ..., valueN) ;
INSERT INTO tab (col1, col2)
  VALUES (value1, value2, value3), (value4, value5, value6) ;
```

Hinweis: es können Werte für mehrere Datensätze übergeben werden.

....