SQL – Tipps und Tricks

PHP-User-Group Stuttgart

09.06.2010

- SQL Tücken im Alltag mit MySQL und anderen Datenbanken
- Performance, Tuning und was man dafür hält..
- praktische Problemlösungen mit SQL
- Lösungen rund um GROUP BY, NULLs und Tuning
- SQL-Standards kennen

SQL – Tipps und Tricks

Wer bin ich ?

Thomas Wiedmann

- > 21 Jahre Problemlösungen in der Softwareentwicklung
- Seit sieben Jahren Projekte mit PHP und Oracle PL/SQL bzw. DB2/NT
- Zend Certified PHP Engineer (ZCE)
- IBM Certified Solution Expert - DB2 UDB v7.1 Database Administration
- Autor diverser Fachartikel in der „Toolbox“ und im PHP-Magazin
- Autor des Buches „DB2 – SQL, Programmierung, Tuning“ © 2001
- Weitere Präsentationen und Tutorials auf meiner Homepage

http://www.twiedmann.de

SQL – Tipps und Tricks

Überblick

- „NULL“ zu „NULL“ Spielereien
- MySQL - GROUP BY oder was man dafür hält..
- MySQL - Server Modus – Überraschung beim Kunden
- Daten „schnell?“ in die Datenbank einlesen
- MySQL – Ein „rasantes“ Ranking erzeugen
- MySQL - Tuning mit GROUP BY
- Design und Datenbank-Migrations Hinweise

Spass mit NULLs und NULL Werten

```
CREATE TABLE test_null (
 wert INT
);

INSERT INTO test_null VALUES
( NULL ), ( NULL );

SELECT COUNT(wert) * 4 AS anzahl FROM test_null;
```

Welcher Wert wird für die Spalte „anzahl“ ermittelt?

a) 1
b) 4
c) 8
d) nichts von alledem

Spass mit NULLs und NULL Werten

```
SELECT COUNT(wert) * 4 AS anzahl FROM test_null;
+---------+
| anzahl  |
+---------+
|       0 |
+---------+
```

Warum ist das so?

COUNT(NULL) * 4 rechnet sich wie „finde nichts“ * 4 = 0

Beim CREATE TABLE also - wenn möglich und sinnvoll - alle Spalten mit dem „NOT NULL“ Constraint definieren.

- spart „Probleme“
- spart „Platz“ („MyISAM“ Engine benötigt bei NULL Spalten ein Byte mehr)

Spass mit NULLs und NULL Werten

```
CREATE TABLE test_null_2 (
 col1 INT NOT NULL,
 col2 INT
);
```

```
INSERT INTO test_null_2 VALUES
 ( 1, NULL );
```

Wie kann exakt dieser Datensatz wieder gelesen werden?

Spass mit NULLs und NULL Werten

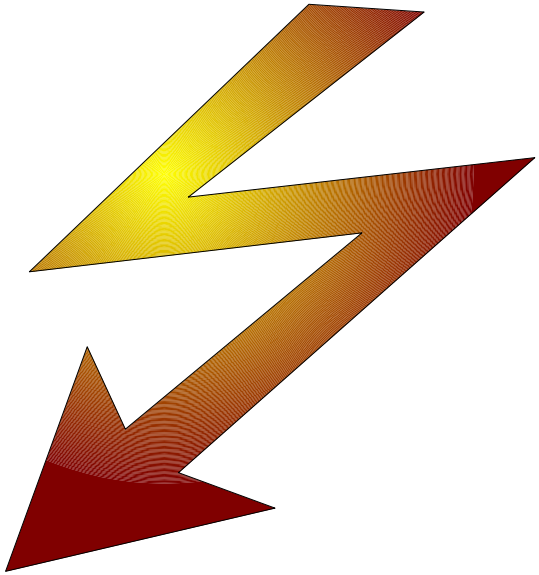
```
<?php
 [...]
 $nCol1 = 1;
 $nCol2 = null;
 $nAnzahl = null;

 $sQuery = "SELECT COUNT(*) AS anzahl FROM test_null_2
            WHERE col1 = ?
              AND col2 = ?";

 if ($stmt = $mysqli->prepare($sQuery)) {

  $stmt->bind_param('ii', $nCol1, $nCol2);
  $stmt->execute();
  $stmt->bind_result($nAnzahl);
  $stmt->fetch();

  $stmt->close();
  echo 'Anzahl='.$nAnzahl;   // liefert => Anzahl=0
 }
[...]
?>
```



Spass mit NULLs und NULL Werten

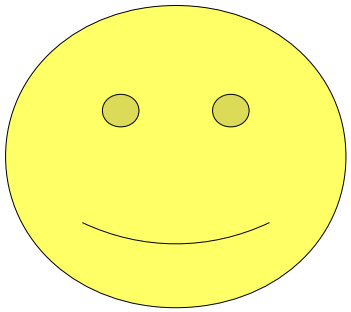
```
<?php
 [...]
 $nCol1 = 1;
 $nCol2 = null;
 $nAnzahl = null;

 $sQuery = "SELECT COUNT(*) AS anzahl FROM test_null_2
            WHERE col1 = ?
              AND col2 IS NULL";

 if ($stmt = $mysqli->prepare($sQuery)) {

  $stmt->bind_param('i', $nCol1);
  $stmt->execute();
  $stmt->bind_result($nAnzahl);
  $stmt->fetch();

  $stmt->close();
  echo 'Anzahl='.$nAnzahl;   // liefert => Anzahl=1
 }
[...]
?>
```



SQL – Tipps und Tricks

Zusammenfassung NULLs

NULL versus **NOT NULL**

- NULL - bedeutet „undefiniert“ bzw. „keine Ahnung“
- NULL - kann nur mit „IS NULL“ korrekt geprüft werden
- NOT NULL - „mehr Probleme“ beim Einfügen und Ändern
- NOT NULL - garantiert bessere Datenqualität
- NULL wird beim ORDER BY unterschiedlich behandelt von den verschiedenen Datenbanken

MySQL - GROUP BY – Er[g | l]ebnisse

```
+----------+--------------+--------+
| kunde_id | vertreter_id | betrag |
+----------+--------------+--------+
|       10 |            1 | 100.00 |
|       10 |            2 | 200.00 |
|       20 |            1 | 300.00 |
|       30 |            2 | 300.00 |
+----------+--------------+--------+
```

```
SELECT kunde_id, vertreter_id, MAX(betrag)
  FROM kunde_umsatz
 GROUP BY kunde_id;
```

```
+----------+--------------+-------------+
| kunde_id | vertreter_id | MAX(betrag) |
+----------+--------------+-------------+
|       10 |            1 |      200.00 |
|       20 |            1 |      300.00 |
|       30 |            2 |      300.00 |
+----------+--------------+-------------+
```

MySQL - GROUP BY – Er[g | l]ebnisse

```
+----------+---------------+--------+
| kunde_id | vertreter_id | betrag |
+----------+---------------+--------+
|       10 |             1 | 100.00 |
|       10 |             2 | 200.00 |
|       20 |             1 | 300.00 |
|       30 |             2 | 300.00 |
+----------+---------------+--------+
```

```
SELECT kunde_id, vertreter_id, MAX(betrag)
  FROM kunde_umsatz
 GROUP BY kunde_id;
```

```
+----------+---------------+--------------+
| kunde_id | vertreter_id | MAX(betrag)  |
+----------+---------------+--------------+
|       10 |             1 |       200.00 |
|       20 |             1 |       300.00 |
|       30 |             2 |       300.00 |
+----------+---------------+--------------+
```

MySQL - GROUP BY – Er[g | l]ebnisse

```
+----------+---------------+--------+
| kunde_id | vertreter_id  | betrag |
+----------+---------------+--------+
|       10 |             1 | 100.00 |
|       10 |             2 | 200.00 |
|       20 |             1 | 300.00 |
|       30 |             2 | 300.00 |
+----------+---------------+--------+
```

```
SELECT kunde_id, vertreter_id, MAX(betrag)
  FROM kunde_umsatz
 GROUP BY kunde_id;
```

```
+----------+---------------+-------------+
| kunde_id | vertreter_id  | MAX(betrag) |
+----------+---------------+-------------+
|       10 |             1 |      200.00 |
|       20 |             1 |      300.00 |
|       30 |             2 |      300.00 |
+----------+---------------+-------------+
```

MySQL liefert „falsche“ Ergebnisse ?!?

MySQL - GROUP BY – Er[g | l]ebnisse

```
+----------+---------------+--------+
| kunde_id | vertreter_id  | betrag |
+----------+---------------+--------+
|       10 |             1 | 100.00 |
|       10 |             2 | 200.00 |
|       20 |             1 | 300.00 |
|       30 |             2 | 300.00 |
+----------+---------------+--------+
```

```
SELECT kunde_id, vertreter_id, MAX(betrag)
  FROM kunde_umsatz
 GROUP BY kunde_id;
```

ungültiger GROUP BY !!

```
+----------+---------------+--------------+
| kunde_id | vertreter_id  | MAX(betrag)  |
+----------+---------------+--------------+
|       10 |             1 |       200.00 |
|       20 |             1 |       300.00 |
|       30 |             2 |       300.00 |
+----------+---------------+--------------+
```

MySQL - GROUP BY – Er[g | l]ebnisse

```
+----------+--------------+--------+
| kunde_id | vertreter_id | betrag |
+----------+--------------+--------+
|       10 |            1 | 100.00 |
|       10 |            2 | 200.00 |
|       20 |            1 | 300.00 |
|       30 |            2 | 300.00 |
+----------+--------------+--------+
```

```
SELECT u.kunde_id, u.vertreter_id, g.max_betrag
  FROM kunde_umsatz u
  JOIN ( SELECT kunde_id, MAX(betrag) AS max_betrag
           FROM kunde_umsatz
          GROUP BY kunde_id ) g
    ON u.kunde_id = g.kunde_id
   AND u.betrag = g.max_betrag
ORDER BY kunde_id;
```

```
+----------+--------------+------------+
| kunde_id | vertreter_id | max_betrag |
+----------+--------------+------------+
|       10 |            2 |     200.00 |
|       20 |            1 |     300.00 |
|       30 |            2 |     300.00 |
+----------+--------------+------------+
```

MySQL - SQL Modus und GROUP BY

Möglichkeiten MySQL „zu zwingen“ eher ANSI-SQL zu verarbeiten.
(durch Anpassen des SQL-Modus des Servers)

- **ONLY_FULL_GROUP_BY**

Nur Spalten der GROUP BY-Klausel dürfen direkt in der Select-Liste stehen. Für alle anderen müssen Aggregatfunktionen wie MIN(), MAX(), COUNT(), SUM() etc. angewendet werden.

- **ANSI**

Ändert Syntax und Verhalten so, dass eine höhere Kompatibilität mit Standard-SQL erzielt wird. ANSI enthält folgende Modi:
REAL_AS_FLOAT, PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE

mehr Details dazu siehe „5.2.5. Der SQL-Modus des Servers“

MySQL - SQL Modus und GROUP BY

MySQL - SQL Modus [global] oder [session] ermitteln

Die Einstellung der GLOBAL-Variable erfordert die Berechtigung SUPER und wirkt sich auf den Betrieb aller Clients aus, die nachfolgend eine Verbindung herstellen. Von der Änderung der SESSION-Variable ist nur der aktuelle Client betroffen. Jeder Client kann jederzeit seinen eigenen sql_mode-Sitzungswert ändern.
© MySQL Handbuch

- **SELECT @@global.sql_mode;**
- **SELECT @@session.sql_mode;**

```
mysql> SELECT @@global.sql_mode;
+--------------------+
| @@global.sql_mode |
+--------------------+
|                    |
+--------------------+
```

MySQL - SQL Modus und GROUP BY

MySQL - SQL Modus [global] und/oder [session] setzen

- SET session sql_mode = 'ONLY_FULL_GROUP_BY';
- SET global sql_mode = 'ANSI';

```
mysql> SELECT kunde_id, vertreter_id, MAX(betrag)
    ->   FROM kunde_umsatz
    ->  GROUP BY kunde_id;
ERROR 1055 (42000): 'testdb.kunde_umsatz.vertreter_id' isn't
in GROUP BY
mysql>
```

wegen sql_mode = 'ONLY_FULL_GROUP_BY'

SQL – Tipps und Tricks

Zusammenfassung GROUP BY

GROUP BY und SQL-Standard

- MySQL erlaubt „falsche“ GROUP BY Konstruktionen
  http://dev.mysql.com/doc/refman/5.0/en/group-by-hidden-columns.html
- SELECT @@global.sql_mode; liefert den Server SQL-Modus
- SELECT @@session.sql_mode; liefert den session SQL-Modus
- sql_mode = 'ONLY_FULL_GROUP_BY' kennen und nutzen
- Mehr zu Server-Modus siehe „5.2.5. Der SQL-Modus des Servers“

MySQL - Überraschungen beim Kunden

```
CREATE TABLE ueberraschung (
 eins CHAR(1) NOT NULL,
 zwei CHAR(1) NOT NULL
);

mysql> SELECT * FROM ueberraschung;
+------+------+
| eins | zwei |
+------+------+
| J    | N    |
| N    | N    |
+------+------+
2 rows in set (0.00 sec)
```

MySQL - Überraschungen beim Kunden

```
SELECT eins || zwei AS wert
  FROM ueberraschung
 ORDER BY wert;
```

Welches Ergebnis liefert die SQL-Abfrage ??

a)

```
+------+
| wert |
+------+
|    0 |
|    0 |
+------+
2 rows in set,
8 warnings
```

b)

```
+--------+
| wert   |
+--------+
| JN     |
| NN     |
+--------+
2 rows in set
```

MySQL - Überraschungen beim Kunden

```
SELECT eins || zwei AS wert
  FROM ueberraschung
 ORDER BY wert;
```

Je nach dem aktiven SQL_MODE a) oder b)

b)
sql_mode = 'PIPES_AS_CONCAT'

(= ANSI-Standard)

a) || = PIPE = OR

```
+------+
| wert |
+------+
|    0 |
|    0 |
+------+
2 rows in set,
8 warnings
```

b) || = STRING CONCAT

```
+--------+
| wert   |
+--------+
| JN     |
| NN     |
+--------+
2 rows in set
```

SQL – Tipps und Tricks

Zusammenfassung SERVER MODUS

MySQL SERVER MODUS und SQL-Standard

MySQL kann in verschiedenen SQL-Modi betrieben werden.
Die SQL-Modi steuern die Kompatibilität zum SQL-Standard und anderen Datenbanken.

5.2.5 Der SQL-Modus des Servers

- ANSI
- DB2
- ORACLE
- TRADITIONAL
- und viele weitere...

Neue Daten schnell einfügen

10.000 neue Adressen in diese Tabelle einlesen:

```
CREATE TABLE adresse (
 id INT NOT NULL,
 name VARCHAR(50) NOT NULL,
 ort VARCHAR(50) NOT NULL,
 telefon VARCHAR(50) NOT NULL,

 PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Einfügen mit PDO::exec() – 1. Versuch

```
<?php
 [...]
 $database = new database();
 $dbh = $database->connect_pdo_mysql();

 if ($dbh) {

  $nStart = microtime(true);

  for($i=1 ; $i < 10000; $i++) {
   $id = $i;
   $sName = 'Nachame-'.$i;
   $sOrt = 'Ort-'.$i;
   $sTelefon = 'Telefon-'.$i;
   $sInsert = "INSERT INTO adresse ( id, name, ort, telefon ) VALUES ";
   $sInsert .= "($id, '$sName', '$sOrt', '$sTelefon' ) ";
   $dbh->exec($sInsert);
  }
  $nEnde = microtime(true);

  $dbh->exec('COMMIT');
  [...]

?>
```

Zeit: ca. 90 Sekunden (PHP 5.2.5 / MySQL 5.1.30)

Einfügen mit PDO::exec() – 2. Versuch

```
<?php
 [...]
 $database = new database();
 $dbh = $database->connect_pdo_mysql();

 if ($dbh) {
  $dbh->beginTransaction();   // <= wichtig bei PDO !!
  $nStart = microtime(true);

  for($i=1 ; $i < 10000; $i++) {
   $id = $i;
   $sName = 'Nachame-'.$i;
   $sOrt = 'Ort-'.$i;
   $sTelefon = 'Telefon-'.$i;
   $sInsert = "INSERT INTO adresse ( id, name, ort, telefon ) VALUES ";
   $sInsert .= "($id, '$sName', '$sOrt', '$sTelefon' ) ";
   $dbh->exec($sInsert);
  }
  $nEnde = microtime(true);

  $dbh->exec('COMMIT');
  [...]

?>
```

Zeit: ca. 0,99 Sekunden (PHP 5.2.5 / MySQL 5.1.30)

Einfügen mit PDO->prepare/execute() – 3. Versuch

```
<?php
 [...]
 $database = new database();
 $dbh = $database->connect_pdo_mysql();
 if ($dbh) {
  $dbh->beginTransaction();
  $nStart = microtime(true);
  $sInsert = 'INSERT INTO adresse ( id, name, ort, telefon ) VALUES ';
  $sInsert .= '(:id, :sName, :sOrt, :sTelefon )';
  $stmt = $dbh->prepare($sInsert);
  $stmt->bindParam(':id',$id, PDO::PARAM_INT);
  $stmt->bindParam(':sName',$sName, PDO::PARAM_STR, 50);
  $stmt->bindParam(':sOrt',$sOrt, PDO::PARAM_STR, 50);
  $stmt->bindParam(':sTelefon',$sTelefon, PDO::PARAM_STR, 50);

  for($i=1; $i < 10000;$i++) {
   $id = $i;
   $sName = 'Nachame-'.$i;
   $sOrt = 'Ort-'.$i;
   $sTelefon = 'Telefon-'.$i;
   $stmt->execute();
  }
?>
```

Zeit: ca. 1,01 Sekunden (PHP 5.2.5 / MySQL 5.1.30)

Einfügen mit LOAD DATA LOCAL – 4. Versuch

```
---adresse.csv----
ID;NAME;ORT;TELEFON
1;"Nachame-1";"Ort-1";"Telefon-1"
2;"Nachame-2";"Ort-2";"Telefon-2"
[…]
```

```
mysql> LOAD DATA LOCAL INFILE 'adresse.csv'
    ->   REPLACE INTO TABLE adresse
    ->           FIELDS TERMINATED BY ';'
    ->           ENCLOSED BY '\"'
    ->           ESCAPED BY '\\'
    ->           LINES TERMINATED BY '\r\n'
    ->           IGNORE 1 LINES;
Query OK, 9999 rows affected (0.14 sec)
Records: 9999  Deleted: 0  Skipped: 0  Warnings: 0

mysql>
```

Zeit: ca. 0,14 Sekunden (MySQL 5.1.30)

SQL – Tipps und Tricks

Zusammenfassung - schnelles Einfügen

Daten können auf verschiedene Arten in die Datenbank eingelesen werden:

MySQL - PDO

a) mit normalen INSERT - ca. 0,99 Sek.
b) prepared Statement: 1 x prepared() und x execute() - ca. 1,01 Sek
c) LOAD DATA (als schnellste Variante) - ca. 0,14 Sek

Prepared/Execute Vergleich - PDO

a) **MySQL** bei a) und b) nahezu identisch (ungewöhnlich)
b) **IBM DB2** bei a) ca. 4,5 Sekunden und b) nur ca. **0,5** Sekunden

allgemeiner Tipp:

Vor dem Einlesen alle Indices dieser Tabelle entfernen (DROP) und anschließend wieder erzeugen (CREATE) oder via
ALTER TABLE tabelle [disable keys | enable keys] deaktvieren | aktivieren

MySQL - Rasantes Ranking erzeugen

Ranglisten und Top-10-Listen jeder Art und Ausprägung sind häufig zu finden. Welches Team ist hier auf Platz 1?

```
CREATE TABLE spiel (
 spiel_id INT NOT NULL,
 team_id INT NOT NULL,
 punkte INT NOT NULL,
 PRIMARY KEY (spiel_id)
);
```

```
INSERT INTO spiel VALUES
( 1, 1, 0),
( 2, 2, 1),
( 3, 3, 0),
( 4, 1, 2),
( 5, 2, 1),
( 6, 3, 2),
( 7, 1, 0),
( 8, 2, 2),
( 9, 2, 0),
(10, 3, 1),
(11, 4, 1),
(12, 4, 1);
```

MySQL - Rasantes Ranking erzeugen

Ranglisten und Top-10-Listen jeder Art und Ausprägung sind häufig zu finden. Welches Team ist hier auf Platz 1?

Gewünschtes Ausgabeergebnis der SQL-Abfrage!

```
+---------+-------+------+
| team_id | summe | rank |
+---------+-------+------+
|       2 |     4 |    1 |
|       3 |     3 |    2 |
|       1 |     2 |    3 |
|       4 |     2 |    3 |
+---------+-------+------+
4 rows in set (0.00 sec)

mysql>
```

MySQL - Rasantes Ranking erzeugen

Ranglisten und Top-10-Listen jeder Art und Ausprägung sind häufig zu finden. Welches Team ist hier auf Platz 1?

team_id	summe	rank
2	4	1
3	3	2
1	2	3
4	2	3

woher?

```
...,SUM(punkte) AS summe
GROUP BY team_id
ORDER BY summe DESC
```

MySQL - Rasantes Ranking erzeugen

MySQL kennt sogenannte User-Variablen (@name), die zum Rechnen genutzt werden können.

```
mysql> SELECT ranking.team_id, ranking.summe,
    ->        @rank := @rank +1 AS rank
    ->   FROM (SELECT @rank := 0) AS init,
    ->        (SELECT team_id, SUM(punkte) AS summe
    ->           FROM spiel
    ->         GROUP BY team_id
    ->         ORDER BY summe DESC) AS ranking;
+---------+-------+------+
| team_id | summe | rank |
+---------+-------+------+
|       2 |     4 |    1 |
|       3 |     3 |    2 |
|       1 |     2 |    3 |
|       4 |     2 |    4 |
+---------+-------+------+
4 rows in set (0.02 sec)

mysql>
```

MySQL - Rasantes Ranking erzeugen

```
mysql> SELECT ranking.team_id, ranking.summe,
    ->        CASE
    ->         WHEN @prev_summe <> ranking.summe
                   THEN @rank := @rank +1
    ->         ELSE @rank
    ->        END AS rank,
    ->        @prev_summe := ranking.summe AS gruppen_test
    ->   FROM (SELECT @rank := 0, @prev_summe := 0) AS init,
    ->        (SELECT team_id, SUM(punkte) AS summe
    ->           FROM spiel
    ->         GROUP BY team_id
    ->         ORDER BY summe DESC) AS ranking;
+---------+-------+------+---------------+
| team_id | summe | rank | gruppen_test |
+---------+-------+------+---------------+
|       2 |     4 |    1 |             4 |
|       3 |     3 |    2 |             3 |
|       1 |     2 |    3 |             2 |
|       4 |     2 |    3 |             2 |
+---------+-------+------+---------------+
4 rows in set (0.09 sec)
```

SQL – Tipps und Tricks

Zusammenfassung – Ranking erzeugen

MySQL und Benutzerdefinierte Variablen wie @rank

Benutzerdefinierte Variablen sind verbindungsspezifisch, d. h., eine Benutzervariable, die von einem Client definiert wurde, wird von anderen Clients nicht gesehen und kann von diesen auch nicht verwendet werden.

Benutzerdefinierte Variablen sind von der Reihenfolge abhängig, wie der SQL abgearbeitet wird. Also testen, testen, testen..!

Im SQL-Standard soll rownum die Reihenfolge analog @rank liefern.

- **ORACLE kennt rownum**
- **IBM DB2 kennt rownum**

Tuning mit GROUP BY

```
CREATE TABLE tuning_group (
 id INT NOT NULL,
 bezeichnung VARCHAR(50)
);

INSERT INTO tuning_group VALUES
( 1, 'Dies ist ein Text..' ),
( 2, 'Dies ist ein..' ),
( 3, 'Dies ist..' ),
( 4, 'Dies..' );
```

Tuning mit GROUP BY

```
mysql> SELECT LENGTH(bezeichnung), COUNT(*)
    ->   FROM tuning_group
    -> GROUP BY LENGTH(bezeichnung);

+----------------------+-----------+
| LENGTH(bezeichnung) | count(*) |
+----------------------+-----------+
|                    6 |         1 |
|                   10 |         1 |
|                   14 |         1 |
|                   19 |         1 |
+----------------------+-----------+
4 rows in set (0.03 sec)
```

Tuning mit GROUP BY

```
mysql> SELECT LENGTH(bezeichnung), COUNT(*)
    ->   FROM tuning_group
    -> GROUP BY LENGTH(bezeichnung);

+----------------------+-----------+
| LENGTH(bezeichnung) | count(*) |
+----------------------+-----------+
|                    6 |         1 |
|                   10 |         1 |
|                   14 |         1 |
|                   19 |         1 |
+----------------------+-----------+
4 rows in set (0.03 sec)
```

sortiert !!

Tuning mit GROUP BY

```
mysql> EXPLAIN
    -> SELECT LENGTH(bezeichnung), COUNT(*)
    ->   FROM tuning_group
    -> GROUP BY LENGTH(bezeichnung);


+----+-------------+-..-+-----------------------------+
| id | select_type | .. | Extra                       |
+----+-------------+-..-+-----------------------------+
|  1 | SIMPLE      | .. | Using temporary; Using filesort |
+----+-------------+-..-+-----------------------------+
1 row in set (0.02 sec)
```

=> MySQL erzeugt auf jeden GROUP BY automatisch einen identischen ORDER BY.

=> Using filesort

„MySQL muss einen zusätzlichen Durchlauf vornehmen, um zu ermitteln, wie die Datensätze in sortierter Reihenfolge abgerufen werden können.“

Tuning mit GROUP BY

```
mysql> EXPLAIN
    -> SELECT LENGTH(bezeichnung), COUNT(*)
    ->   FROM tuning_group
    -> GROUP BY LENGTH(bezeichnung)
    -> ORDER BY NULL;

+----+-------------+-..-+-----------------+
| id | select_type | .. | Extra           |
+----+-------------+-..-+-----------------+
|  1 | SIMPLE      | .. | Using temporary |
+----+-------------+-..-+-----------------+
1 row in set (0.02 sec)
```

=> Mit ORDER BY NULL die „Defaultsortierung“ deaktivieren
=> kein Using filesort mehr notwendig

SQL – Tipps und Tricks

Zusammenfassung - Tuning mit GROUP BY

Beispiel:

```
SELECT k.name, u.letzt_verkauf
  FROM kunde k
  JOIN (SELECT kunde_id, MAX(verkauf) AS letzt_verkauf
          FROM kunde_umsatz
        GROUP BY kunde_id
        ORDER BY NULL) u
    ON u.kunde_id = k.kunde_id
ORDER BY k.name;
```

Die Defaultsortierung innerhalb dem SUB-Select u ist unnötig, da das Gesamtergebnis eine andere Sortierung erhält.

Design und Datenbank-Migrations Hinweise

```
CREATE TABLE Schüler (
 S_id INT NOT NULL,
 Vorname VARCHAR(20) NOT NULL,
 Nachname VARCHAR(30) NOT NULL,
 Geburtstag DATE NOT NULL,
 alter INT NOT NULL,
 PRIMARY KEY(S_id)
);
```

Was ist hier nicht ideal bzw. nicht korrekt..?

Design und Datenbank-Migrations Hinweise

```
CREATE TABLE `Schüler` (
 `S_id` INT NOT NULL,
 `Vorname` VARCHAR(20) NOT NULL,
 `Nachname` VARCHAR(30) NOT NULL,
 `Geburtstag` DATE NOT NULL,
 alter INT NOT NULL,
 PRIMARY KEY(`S_id`)
);
```

Der Feld-, Wald- und Wiesentipp bei MySQL-Design-problemen sind Ticks!

Funktioniert der CREATE TABLE jetzt?

Design und Datenbank-Migrations Hinweise

```
CREATE TABLE `Schüler` (
 `S_id` INT NOT NULL,
 `Vorname` VARCHAR(20) NOT NULL,
 `Nachname` VARCHAR(30) NOT NULL,
 `Geburtstag` DATE NOT NULL,
```

~~alter INT NOT NULL,~~

```
 PRIMARY KEY(`S_id`)
);
```

Reserved Word-Problem und Design-Problem!

Funktioniert der CREATE TABLE jetzt?

Design und Datenbank-Migrations Hinweise

```
mysql> CREATE TABLE `Schüler` (
    ->  `S_id` INT NOT NULL,
    ->  `Vorname` VARCHAR(20) NOT NULL,
    ->  `Nachname` VARCHAR(30) NOT NULL,
    ->  `Geburtstag` DATE NOT NULL,
    ->  PRIMARY KEY(`S_id`)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO `Schüler` VALUES
    -> ( 1, 'Fritz', 'Frech', '1995-01-01'),
    -> ( 2, 'Else', 'Elster', '1996-01-01'),
    -> ( 3, 'Otto', 'Obacha', '1997-01-01');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Funktioniert! Ist das aber wirklich so toll?

Design und Datenbank-Migrations Hinweise

```
mysql> show tables like "Sch%";
+--------------------------+
| Tables_in_sample (Sch%)  |
+--------------------------+
| sch?ler                  |
+--------------------------+
1 row in set (0.00 sec)

mysql>
```

Probleme mit dem Zeichensatz

```
<?php
  [...]
  /* fetch associative array */
  while ($row = $result->fetch_assoc()) {

   echo $row['S_id'];
   echo $row['Vorname'];

  }
?>
```

case-sensitive Elementnamen!!

SQL – Tipps und Tricks

Zusammenfassung

- keine Umlaute und Sonderzeichen
  auf Datenbank- und Programmebene verwenden
- keine „Tricks“ mit den MySQL-Ticks
- andere Datenbanken liefern die Spaltennamen
  per Default in Grossbuchstaben
- reservierte Worte kennen
- SQL-Standard berücksichtigen
- mal über den eigenen Datenbank-Tellerrand schauen

SQL – Tipps und Tricks

Propritäre Lösungen
optimale Performance

„einfache“ Lösung
(Framework)

SQL Standard

SQL – Tipps und Tricks

Propritäre Lösungen
optimale Performance

The End!

SQL im

Spannungsfeld

zwischen

„einfache“ Lösung
(Framework)

SQL Standard