Fehlerbehandlung

- Replikationsfehler Errno: 1236
- Replikationsfehler Errno: 1580 ALTER Logtable
- Replikationsfehler Errno: 1594 Relay log read failure
- Fehler bei ALTER TABLE Errno 150: Foreign key constraint is incorrectly formed
- InnoDB - defekte FRM Files wiederherstellen - Incorrect information in file
- Probleme mit Umlauten zwischen Datenbank und Webseite (UTF-8/latin1/ISO8859-1)

Replikationsfehler Errno: 1236

Replikationsfehler Errno: 1236: start replication from impossible position

```
Slave I/O: Got fatal error 1236 from master when reading data from binary log: 'Client
requested master to start replication from impossible position; the first event mysql-
bin.010346 at 5134523, the last event read from /var/lib/mysql/mysql-bin.000346 at 4, the last
byte read from /var/lib/mysql/mysql-bin.010346 at 4., Error_code: 1236
```

Dieser Fehler kann mehrere Ursachen haben. Die häufigste Ursache bei meinen Server war ein harter Crash des Master-Servers. Dabei wurden die letzten Binlog-Einträge nicht auf die Disk geschrieben oder es landete Müll darin.

Das kann man herausfinden, wenn man sich das letzte Binlog (auf dem Master) ansieht. Sollte das Log hier zu Ende sein, muss man dem Slave das darauf folgende Logfile und die neue Logposition mitteilen.

```
$ mysqlbinlog --no-defaults --base64-output=decode-rows --verbose --verbose --start-
position=5134523 mysql-bin.010346
```

Das nächste Logfile wäre in diesem Fall /var/lib/mysql/mysql-bin.000347, bei Position 4.

```
mysql> STOP SLAVE;
mysql> CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000347', MASTER_LOG_POS=4;
mysql> START SLAVE;
```

Die Replikation sollte nun wieder normal weiterlaufen.

Um dem Fehler entgegen zui wirken, kann man „sync_binlog = 1“ in der my.cnf im [mysqld]-Bereich setzen. Damit führt MySQL nach jedem commit einen fsync auf der Disk aus. Bei entsprechend schnellen Platten bedeutet diese Option nur einen minimalen Performanceverlust.

Replikationsfehler Errno: 1580 ALTER Logtable

Nach einem Update von MySQL bei einer Master-Slave-Replikation konnte die Replikation auf einem Slave-Server kürzlich nicht mehr gestartet werden.

```
Last_Errno: 1580
Last_Error: Error 'You cannot 'ALTER' a log table if logging is enabled' on query. Default
database: 'mysql'. Query: 'ALTER TABLE slow_log
```

Offenbar hat der Slave Probleme, das „`ALTER TABLE`“ anzuwenden, sofern das Logging noch aktiv ist (in diesem Fall das Slow-Query Log).

Der Fehler läßt sich relativ einfach beheben (über die MySQL CLI):

```
STOP SLAVE;
SET GLOBAL slow_query_log = 'OFF';
START SLAVE;
SHOW SLAVE STATUS\G
```

Der Fehler sollte nun verschwunden sein, der Slave konnte die Query also auf seine DB anwenden. Nun kann auch das Slow-Query Log wieder aktiviert werden:

```
mysql> SET GLOBAL slow_query_log = 'ON';
```

Replikationsfehler Errno: 1594 Relay log read failure

Wenn ein Server, auf dem ein MySQL-Replication-Slave läuft abstürzt, kann es passieren, dass die Master-Slave-Replikation nicht mehr funktioniert. Beim Absturz landen korrupte Daten in den Relay-Logs, die von MySQL nach dem Neustart nicht verarbeitet werden können. Im SLAVE STATUS steht dann folgendes:

Last_Errno: 1594
Last_Error: Relay log read failure: Could not parse relay log event entry. The possible reasons are: the master's binary log is corrupted (you can check this by running 'mysqlbinlog' on the binary log), the slave's relay log is corrupted (you can check this by running 'mysqlbinlog' on the relay log), a network problem, or a bug in the master's or slave's MySQL code. If you want to check the master's binary log or slave's relay log, you will be able to know their names by issuing 'SHOW SLAVE STATUS' on this slave.
In diesem Fall reicht es häufig die Relay-Logs vom Slave zu löschen und die Replikation neu zu starten. Ein erneuter Dump der Datenbank ist dabei nicht notwendig. Wir sagen dem Server einfach, wo er unterbrochen wurde und setzen die Replikation an dieser Stelle neu auf.

1. Slave-Prozesse stoppen: `STOP SLAVE;`
2. Aus `SLAVE STATUS` die folgenden Werte merken: „Relay_Master_Log_File“ und „Exec_Master_Log_Pos“
3. Slave zurücksetzen, dabei werden die korrupten Logs gelöscht: `RESET SLAVE;`
4. jetzt sagen wir dem Slave an welcher Stelle die Replikation unterbrochen wurde mit den Werten aus dem zweiten Schritt. Master-Host, Username usw. müssen wir dabei nicht nochmal eingeben: `CHANGE MASTER TO MASTER_LOG_FILE='Relay_Master_Log_File', MASTER_LOG_POS=Exec_Master_Log_Pos;`
5. nun kann der Slave wieder gestartet werden und sollte an der richtigen Stelle weitermachen. Das zuvor korrupte Binlog wird erneut vom Master übertragen: `START SLAVE;`

Fehler bei ALTER TABLE Errno 150: Foreign key constraint is incorrectly formed

Dieser Fehler trat bei einem „`ALTER TABLE \`table\` ENGINE = InnoDB`“ auf (die Tabelle sollte von MyISAM auf InnoDB umgestellt werden). Eine Spalte war von einer anderen Tabelle (bereits InnoDB) referenziert. Nach dem der Contraint von der anderen Tabelle entfernt worden war, ließ sich die Tabelle konvertieren. Anschließend kann der Constraint wieder hinzugefügt werden.

Mit diesem Query lassen sich Referenzen von externen Tabellen auf die zu konvertierende Tabelle anzeigen:

```
SELECT
  TABLE_NAME,COLUMN_NAME,CONSTRAINT_NAME, REFERENCED_TABLE_NAME,REFERENCED_COLUMN_NAME
FROM
  INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
  REFERENCED_TABLE_SCHEMA = '<database>' AND
  REFERENCED_TABLE_NAME = '<table>';
```

InnoDB - defekte FRM Files wiederherstellen - Incorrect information in file

Sollten InnoDB-Datenbanken korrupt sein und auch keinen Dump mehr erlauben, da diese mit folgendem Fehler abbrechen „Incorrect information in file: ‘<table name> .frm“, kann man sich vielleicht noch mit diesem Vorgehen helfen (InnoDB erlaubt auch kein „repair tables“):

File Backup erstellen:

```
# /etc/init.d/mysqld stop
# mkdir /root/mysql_backup
# cp -rv /var/lib/mysql/* /root/mysql_backup/
```

In die MySQL Konfigurationsdatei my.cnf unter [mysqld] innodb_force_recovery = 4 setzen und die Datenbank wieder starten via:

```
mysqldump -uroot -pPASSWORD -A > dump.sql
```

einen Dump erstellen, sollte dieser fehlschlagen, stoppen wir die Datenbank und erhoehen den innodb_force_recovery auf 5 oder im naechsten Schritt auf 6. Laeuft der Dump durch, wird die Datenbank gestoppt und alle Datenbanken und Tabellen geloescht:

```
rm -rf /var/lib/mysql/*
```

Nun wird innodb_force_recovery in der my.cnf auskommentiert und die Datenbank wieder gestartet. Jetzt koennen wir die Datenbanken wieder einspielen:

```
mysql -uroot -pPASSWORD < dump.sql
```

Nachdem die Datenbanken nun wieder laufen erstellen wir uns einen Cronjob fuer das taegliche Sichern aller Datenbanken.

Probleme mit Umlauten zwischen Datenbank und Webseite (UTF-8/latin1/ISO8859-1)

Bei der Wiederherstellung oder dem Umzug einer Datenbank kann es manchmal zu Problemen kommen. Sonderzeichen und Umlaute werden dann auf der Webseite nicht mehr korrekt dargestellt. Eine Ursache ist z.B. wenn auf dem alten Server die Datenbank als Latin1 gedumpt wird und auf dem neuen ohne vorherige Konvertierung in eine UTF-8 DB importiert wird. MySQL und PHP gehen dann davon aus, dass die Daten in UTF-8 sind, obwohl diese eigentlich im Latin1 Zeichensatz vorliegen.

Dieses Problem läßt sich wie folgt beheben.

Dump der Datenbank, wir sagen dem mysqldump explizit, dass es sich um Latin1 handelt:

```
mysqldump --skip-extended-insert --default-character-set=latin1 <database> -r <database>-latin1.sql
```

- -skip-extended-insert fasst mehrere Zeilen nicht mehr in einem einzelnen INSERT zusammen. Das erleichtert einen Diff, verlangsamt allerdings bei großen Datenbanken den Reimport deutlich (optional)
- -r speichert den Output direkt in eine Datei. Es soll schon vorgekommen sein, dass durch die Umleitung von STDOUT über > in eine Datei komische Sachen passiert sind

Mit dem Tool iconv wird nun das Dumpfile tatsächlich auf UTF-8 gesetzt. Der Parameter „-c“ ist dabei optional. Er läßt Zeichen, die nicht konvertiert werden können einfach weg (Achtung: möglicher Datenverlust).

```
iconv -f UTF-8 -t UTF-8 -c <database>-latin1.sql > <database>-utf8.sql
```

Wenn wir nun die beiden Dateien vergleichen, sollten in der utf8-Datei die Umlaute korrekt angezeigt werden:

```
diff <database>-latin1.sql <database>-utf8.sql
```

Im Dumpfile müssen eventuelle Datenbanken/Tabellen noch auf UTF-8 gestellt werden:

```
# grep latin1 <database>-utf8.sql
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
  `name` varchar(255) CHARACTER SET latin1 NOT NULL,
  `description` text CHARACTER SET latin1 NOT NULL,
  `source` varchar(255) CHARACTER SET latin1 NOT NULL,


# sed -i 's/latin1/utf8/g' <database>-utf8.sql
```

Der MySQL-Server und die verbindenden Clients sollten nun natürlich auch UTF-8 liefern/schreiben. Dazu brauchts ein paar Ergänzungen an den entsprechenden Stellen in der my.cnf:

```
[client]
[...]
default-character-set   = utf8


[...]


[mysqld]
[...]
character-set-server    = utf8
collation-server        = utf8_unicode_ci
```

Danach den MySQL-Server neu starten (Achtung: Das nachträgliche Umstellen des Zeichensatzes kann unter Umständen bei bestehenden Datenbanken für Probleme sorgen).

```
# service mysql restart
```

Die Einstellungen überprüfen:

```
mysql> show variables like '%character_set_%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)
```

Nun kann man den Dump einspielen:

```
mysql <datenbank> < <database>-utf8.sql
```