Fourth Annual

MySQL®

Users Conference 2006

MySQL 5.1

Past, Present and Future

Jan Kneschke
MySQL AB

Presented by MySQL O'REILLY

Agenda

- Past
  - SQL Trees meets Dynamic SQL
- Present
  - Events
  - Partitioning
- Future
  - Vertical Partitioning

About the Presenter

- Jan Kneschke
  - Senior Developer in the MySQL Network Team
  - jan@mysql.com
- Drives the development of the high-performance web-server **lighttpd**
- Was trainer and consultant at MySQL before becoming a developer

Raise your hands

- Who is using MySQL 4.1 or older ?
- Who is using MySQL 5.0, 5.1 or later ?
- Who uses
  - Prepared Statements
  - Stored Procedures
  - Partitioning
- Who is using one of the features with another vendors RDBMS ?

Back in Time

- Last years tutorial concentrated on
  - Stored Procedures, VIEWs and Triggers
- Examples can be found at
  http://jan.kneschke.de/projects/mysql/sp/
- Who has attended last years tutorial ?

Prepared Statements

- First available in MySQL 4.1
- Split up the execution of SQL statements into a PREPARE and a EXECUTE phase
- At best PREPARE once, EXECUTE multiple times
- Prevents SQL-injection

Prepared Statements

```
PREPARE s FROM 'SELECT * FROM tbl
 WHERE id = ?';

EXECUTE s USING @id;

DEALLOCATE PREPARE s;
```

Stored Procedures

- A programming language running in the context of the DBMS
- Uses the syntax defined in the SQL:2003 standard
- Provides Control-Flow, Loops, Exceptions, Cursors, ...
- SPs are used by Triggers, Events, Functions

Stored Procedures

```
CREATE PROCEDURE fill_table ()
BEGIN
  DECLARE n INT DEFAULT 1000;
  ins_loop: LOOP
    INSERT INTO tbl VALUES (n);
    SET n = n - 1;
    IF (NOT n) THEN
      LEAVE ins_loop;
    END IF;
  END LOOP ins_loop;
END$$
```

Dynamic SQL

- fill_table() works only against one table
- SQL Statements in SPs are static
- Prepared Statements can only use placeholders for values, not for parts of the SQL statement itself
- PREPARE takes a constant string or (since MySQL 5.0.13) a user variable

Dynamic SQL

```
SET @s = CONCAT(
  "INSERT INTO ", tbl_name,
  " (", field_name,") VALUES (?)");
PREPARE s FROM @s;
SET @n = n;
EXECUTE s USING @n;
DEALLOCATE PREPARE s;
```

SQL Trees

- New implementation of SQL trees using Dynamic SQL
- Only requirement
  - `id INT NOT NULL PRIMARY KEY` as node-id
  - `parent_id INT NOT NULL` referencing the node-id of the parent node
- Name of the source-table and temporary result tables are passed as parameters

SR-lib

- Standard Routines Library
- http://savannah.nongnu.org/projects/mysql-sr-lib/
- A community effort to create a standard library for stored procedures
- Arrays, For-Each loops, syntax helpers, UnitTests
- Example:
  - `CALL for_each_table(<db>, <cond>, <statement>)`

Events

- Events can execute stored procedures at a specific time or in a interval
- They provide the same functionality as `cron` and `at` on UNIX or the task-scheduler on Windows
- Either one-shot or repeating events
- Repeating events can have a end-time which also might remove the event

Events

- The CREATE EVENT statement binds a stored procedure to a event

```
CREATE EVENT ev_name
  ON SCHEDULE
    AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
(or) EVERY 1 WEEK STARTS '00:01:00' ENDS ...
DO <sp>
```

CREATE EVENT

- One Shot events

```
AT <timestamp>
```

- Repeating events

```
EVERY <INTERVAL> [STARTS <timestamp>] [ENDS
  <timestamp>]
```

- Self-Removing events

```
ON COMPLETION [NOT] PRESERVES
```

CREATE EVENT

- Make sure that the statements of the SP is not returning any result sets
  - No SELECT statements which are not SELECT ... INTO or assigned to a variable
  - Event is executed without a client to send the result set to
  - Write the stored procedure without the event first and test it

Testing events

- Test the SP before adding it to a event

```
CREATE PROCEDURE sp_check_tables () ...
```

- Create a one-shot event before activating the event forever

```
CREATE EVENT ev_check_tables
  ON SCHEDULE AT CURRENT_TIMESTAMP
  DO CALL sp_check_tables()
```

The event scheduler

- The event scheduler is triggering the stored procedure according to your event definition
- To check if your MySQL release if having the event-scheduler compiled in

```
SHOW VARIABLE LIKE 'event%';
```

- In 5.1.7 it is disabled by default

```
SET GLOBAL event_scheduler = 1;
```

Disabling events

- In case you want to disable events you can either disable a single event or all

```
SET GLOBAL event_scheduler = 0;

ALTER EVENT ev_check_tables DISABLE;
```

- If you are entering a maintenance mode and want to disable a set of events for a while
  - INFORMATION_SCHEMA.events knows all events
  - Create a SP to disable them

Managing Events

- ALTER EVENT allows you to change most aspects of the CREATE EVENT statement
- DROP EVENT removes the event
- Only the definer (creator) of the event can alter it
- If you have write access to mysql.event you can circumvent the checks of ALTER EVENT – DON'T DO IT

Break

- Coming up
  - Partitioning
- Starting at 3:30

Partitioning

- Horizontal Partitioning splits the a table by rows
- Vertical Partitioning moves columns to other tables
- MySQL 5.1 supports Horizontal Partitioning

Partitioning

- Reasons to use partitioning
  - Speed
  - Easier Management
- Distribution of rows is controlled by a partitioning expression
- The MySQL Cluster was using Partitioning since the first release

Partitioning Expressions

- Defines
  - Number of Partitions
  - Where to store rows for a table
  - How to find rows

```
PARTITION BY RANGE( YEAR(added) ) (
    PARTITION y2004 VALUES LESS THAN (2005),
    PARTITION y2005 VALUES LESS THAN (2006),
    PARTITION y2006 VALUES LESS THAN MAXVALUE
);
```

Partitioning

- Use MySQL 5.1.7 for testing

```
show variables like 'have_par%';
```

- Partitioning works with all Storage Engines
- All Partitions of a table of to be of the same type
  - No mixing of MyISAM and InnoDB

Creating Partitions

```
CREATE TABLE part_key (
   id INT NOT NULL AUTO_INCREMENT PRIMARY
   KEY,
   name VARCHAR(32),
   added DATETIME NOT NULL
) ENGINE = myisam
   PARTITION BY KEY()
   PARTITIONS 2;
```

Expressions

- KEY splits by PRIMARY KEY
- HASH hashes a field with MD5 or PASSWORD
- RANGE creates a partition for a range of values
- LIST creates a partition for each set of values

KEY

- KEY is best used if you want to split the table to utilize multiples disk better
- Good for large tables which have to be split up as they are larger than a single disk
- The simplest way to create a partitioned table

HASH

- HASH is a more generic version of KEY
- The hashed field is provided by the user and might be based on the result of a function
- The hash-function has to return a integer
- The hash-result is taken modulo the number of partitions

```
... HASH( WEEKDAY( ins_date ) )
```

LINEAR HASHing

- Linear Hashing is distribution the hash-result using a quadratic function instead of a direct modulo
- The distribution is supposed to be better if you want to drop partitions later
- Ask Mikael for more

RANGE

- A tighter control of the values in a partition
- A range is define as LESS THAN (<value>)
- If the last range has to catch all rows which are not matched yet it uses MAXVALUE as upper end
- Usually RANGE by year, month, ...

LIST

- List is a set of values for each partition
- If a value is not in one of the lists the INSERT/UPDATE statement fails with an error

Partition Pruning

- READs from a partition only take those partitions into account which pass the WHERE clause
- This is called pruning
- If you only have to scan 1 of 4 partitions the query is executed 4 times faster

Partition Pruning

- EXPLAIN got extended to show pruning

```
EXPLAIN PARTITIONS

   SELECT * FROM part_range

    WHERE added = now()\G

...

   partitions: y2006

...
```

Managing Partitions

- ALTER TABLE is used to change the layout of partitions
  - ADD PARTITION adds a new partition
  - DROP PARTITION drops the partition including its data

Maintaining Partitions

- ALTER TABLE ... REORGANIZE is used to change the partition layout without loosing any data
  - Usually used to change the partition rules for RANGE and LIST

Maintaining Partitions

- ALTER TABLE ... COALESCE reduces the number of partitions for HASH and KEY partitions
- ADD PARTITION is used to increase them
- REORGANIZE doesn't work with HASH and KEY

Maintaining Partitions

- ALTER TABLE ... COALESCE reduces the number of partitions for HASH and KEY partitions
- ADD PARTITION is used to increase them
- REORGANIZE doesn't work with HASH and KEY

Maintaining Partitions

- ALTER TABLE also provides access to OPTIMIZE, ANALYSE, CHECK and REPAIR partitions
- A REBUILD is recreating partitions from scratch
  - All data is dropped and reinserted
  - This can remove fragmentation

Limitations

- No FULLTEXT
- No GEOMETRY columns
- No FOREIGN KEYS
- If a table has a PK all columns of the partitioning expression have to part of the PK

Vertical Partitioning

- Split the table by column instead of by row
- Move large, less used columns into a separate table
- Reducing the average length of a row
- Speed of Queries might go up as less data is read and data can be kept in memory most of the time

Vertical Partitioning

- MySQL doesn't support vertical partitioning by itself
- You have to emulate it in your application
- Perhaps you will see stored procedures helping you with this task in next years tutorial

Gimmicks

```
SELECT user, host, info, time
  FROM INFORMATION_SCHEMA.processlist
 WHERE command = 'locked';
SELECT * FROM mysql.general_log;
```

Gimmicks

Additional Talks

- Dynamic SQL by Konstantin Osipov
- Events by Andrey Hristov
- Writing Advisors for MySQL Network

Thanks

- More questions ?
- Thanks for your attention

Send further questions to

jan@ mysql.com