Using Stored Routines for MySQL Administration

Giuseppe Maxia,
giusepppe@mysql.com
QA Developer, MySQL AB
Giuseppe Maxia

- MySQL QA Developer
- Based in Sardinia, Italy
Giuseppe Maxia
according to Google Maps, excellent swimmer
Giuseppe Maxia

according to Google Maps, excellent swimmer
Agenda

• Overview
• Stored routines language
• Execution environment
• Stored routines and INFORMATION_SCHEMA
• Using dynamic SQL
• Event scheduler
• Stored routines compared to client side languages
• Resources
• Q&A
Agenda

• **Overview**
• Stored routines language
• Execution environment
• Stored routines and INFORMATION_SCHEMA
• Using dynamic SQL
• Event scheduler
• Stored routines compared to client side languages
• Resources
• Q&A
MySQL Stored routines Overview
MySQL Stored routines Overview

- Stored on server
MySQL Stored routines Overview

- Stored on server
- Belong to database
MySQL Stored routines Overview

- Stored on server
- Belong to database
- Based on standard SQL specification
Agenda

• Overview
• **Stored routines language**
• Execution environment
• Stored routines and INFORMATION_SCHEMA
• Using dynamic SQL
• Event scheduler
• Stored routines compared to client side languages
• Resources
• Q&A
MySQL Stored routines language
MySQL Stored routines language

- PROCEDURE
MySQL Stored routines language

- PROCEDURE
- FUNCTION
MySQL Stored routines language

- PROCEDURE
- FUNCTION
- TRIGGER
MySQL Stored routines language - procedure
MySQL Stored routines language - procedure

- PARAMETERS
MySQL Stored routines language - procedure
• PARAMETERS
  • IN
MySQL Stored routines language - procedure

- PARAMETERS
  - IN
  - OUT
MySQL Stored routines language - procedure

- PARAMETERS
  - IN
  - OUT
  - INOUT
MySql Stored routines language - procedure

• PARAMETERS
  • IN
  • OUT
  • INOUT

• may return one or more data sets
MySQL Stored routines language - procedure

- PARAMETERS
  - IN
  - OUT
  - INOUT

- may return one or more data sets
- can use dynamic SQL
MySQL Stored routines language - function
MySQL Stored routines language - function

- only input PARAMETERS
MySQL Stored routines language - function

- only input PARAMETERS
- MUST return one value of a given type
MySQL Stored routines language - function

- only input PARAMETERS
- MUST return one value of a given type
- can't use dynamic SQL
MySQL Stored routines language - function

- only input PARAMETERS
- MUST return one value of a given type
- can't use dynamic SQL
- can't return data sets
MySQL Stored routines language - trigger
MySQL Stored routines language - trigger

- associated to table events (INSERT, UPDATE, DELETE)
MySQL Stored routines language - trigger

- associated to table events (INSERT, UPDATE, DELETE)
- its parameters depend on the event
MySQL Stored routines language - trigger

- associated to table events (INSERT, UPDATE, DELETE)
- its parameters depend on the event
- does not return anything
MySQL Stored routines language - trigger

- associated to table events (INSERT, UPDATE, DELETE)
- its parameters depend on the event
- does not return anything
- can't use dynamic SQL
MySQL Stored routines language - trigger

- associated to table events (INSERT, UPDATE, DELETE)
- its parameters depend on the event
- does not return anything
- can't use dynamic SQL
- can't return data sets
MySQL Stored routines language
MySQL Stored routines language

- CREATE PROCEDURE
MySQL Stored routines language

- CREATE PROCEDURE
- CREATE FUNCTION
MySQL Stored routines language

- CREATE PROCEDURE
- CREATE FUNCTION
- BEGIN .. END blocks
MySQL Stored routines language

- CREATE PROCEDURE
- CREATE FUNCTION
- BEGIN .. END blocks
- IF ... THEN ... ELSE ... END IF
MySQL Stored routines language

- CREATE PROCEDURE
- CREATE FUNCTION
- BEGIN .. END blocks
- IF ... THEN ... ELSE ... END IF
- CASE ... THEN ... THEN ... ELSE ... END CASE
MySQL Stored routines language

- CREATE PROCEDURE
- CREATE FUNCTION
- BEGIN .. END blocks
- IF ... THEN ... ELSE ... END IF
- CASE ... THEN ... THEN ... ELSE ... END CASE
- WHILE ... END WHILE
MySQL Stored routines language

- CREATE PROCEDURE
- CREATE FUNCTION
- BEGIN .. END blocks
- IF ... THEN ... ELSE ... END IF
- CASE ... THEN ... THEN ... ELSE ... END CASE
- WHILE ... END WHILE
- LOOP ... END LOOP
MySQL Stored routines language

- CREATE PROCEDURE
- CREATE FUNCTION
- BEGIN .. END blocks
- IF ... THEN ... ELSE ... END IF
- CASE ... THEN ... THEN ... ELSE ... END CASE
- WHILE ... END WHILE
- LOOP ... END LOOP
- DECLARE
MySQL Stored routines language
MySQL Stored routines language

- HANDLERS
MySQL Stored routines language

- HANDLERS
- CURSORTS
MySQL Stored routines language - procedure
CREATE PROCEDURE how_is_it (IN x INT)
BEGIN
    IF (x > 5) THEN
        SELECT CONCAT(x, " is higher") as answer;
    ELSE
        SELECT CONCAT(x, " is lower") as answer;
    END IF;
END
MySQL Stored routines language - procedure
MySQL Stored routines language - procedure

CALL how_is_it(6);
+---------------------+
| answer              |
+---------------------+
| 6 is higher         |
+---------------------+

CALL how_is_it(2);
+---------------------+
| answer              |
+---------------------+
| 2 is lower          |
+---------------------+
MySQL Stored routines language - function
CREATE FUNCTION is_bigger (x INT) RETURNS CHAR(3) BEGIN
    IF (x > 5) THEN
        RETURN 'YES';
    ELSE
        RETURN 'NO';
    END IF;
END
MySQL Stored routines language - function
MySQL Stored routines language - function

```sql
SELECT is_bigger(6);
+----------------+
| is_bigger(6)   |
+----------------+
| YES            |
+----------------+

SELECT is_bigger(2);
+----------------+
| is_bigger(2)   |
+----------------+
| NO             |
+----------------+
```
MySQL Stored routines language - triggers
CREATE TRIGGER salary_bi
BEFORE INSERT ON salary
FOR EACH ROW
BEGIN
  CASE
    WHEN new.work_done > 10
      THEN SET new.bonus = 5000;
    WHEN new.work_done > 5
      THEN SET new.bonus = 2500;
    WHEN new.work_done > 2
      THEN SET new.bonus = 1000;
    ELSE
      SET new.bonus = 0;
  END CASE;
END
MySQL Stored routines language - triggers
MySQL Stored routines language - triggers

```sql
insert into salary (emp_no, work_done, bonus) values (1, 15, 0);
Query OK, 1 row affected (0.00 sec)

insert into salary (emp_no, work_done, bonus) values (2, 5, 0);
Query OK, 1 row affected (0.00 sec)

select emp_no, work_done, bonus from salary;
+--------+-----------+-------+
<table>
<thead>
<tr>
<th>emp_no</th>
<th>work_done</th>
<th>bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>5000</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1000</td>
</tr>
</tbody>
</table>
+--------+-----------+-------+
```

MySQL Stored routines language - triggers
MySQL Stored routines language - cursors

- variables for cursor columns
- flag variable
- cursor
- continue handler
MySQL Stored routines language - cursors

open

loop

fetch

variables

check flag

process data

handler sets

flag variable

close
MySQL Stored routines language - cursors
DECLARE done BOOLEAN DEFAULT FALSE;
DECLARE my_bonus INT;
DECLARE get_it CURSOR FOR
  SELECT bonus FROM salary;
DECLARE CONTINUE HANDLER FOR SQLSTATE NOT FOUND
  SET done = TRUE;

OPEN CURSOR get_it;
GETTING:
  LOOP
    FETCH get_it INTO my_bonus;
    IF done THEN LEAVE GETTING; END IF
    # do something with my_bonus
  END LOOP;
CLOSE get_it;
Agenda

- Overview
- Stored routines language
- **Execution environment**
- Stored routines and INFORMATION_SCHEMA
- Using dynamic SQL
- Event scheduler
- Stored routines compared to client side languages
- Resources
- Q&A
Execution environment
Execution environment

- DEFINER
Execution environment

- DEFINER
- SQL MODE stored at definition time
Execution environment

• DEFINER

• SQL MODE stored at definition time

• SESSION VARIABLES shared with the external environment
Execution Environment
Execution Environment

SET SQL_MODE = 'STRICT_ALL_TABLES';

CREATE PROCEDURE put (e int, w int)
    INSERT INTO salary (emp_no, work_done)
    values (e,w);
ALTER TABLE salary MODIFY emp_no TINYINT;

SELECT definer, sql_mode,security_type
FROM information_schema.routines
WHERE routine_name = 'put'
    AND routine_schema = SCHEMA();

    definer: gmax@%
sql_mode: STRICT_ALL_TABLES
security_type: DEFINER
Execution Environment
Execution Environment

```
SET SQL_MODE = '';

INSERT INTO SALARY (emp_no, work_done) VALUES (1, 10);
Query OK, 1 row affected (0.00 sec)

call put(1, 10);
Query OK, 1 row affected (0.00 sec)
```
Execution Environment
Execution Environment

```sql
SET SQL_MODE = '';

INSERT INTO SALARY (emp_no, work_done) VALUES (300, 10);
Query OK, 1 row affected, 1 warning (0.00 sec)
Warning (Code 1264): Out of range value adjusted for column 'emp_no' at row 1

select emp_no, work_done, bonus from salary;
```

<table>
<thead>
<tr>
<th>emp_no</th>
<th>work_done</th>
<th>bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>2500</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>2500</td>
</tr>
<tr>
<td>127</td>
<td>10</td>
<td>2500</td>
</tr>
</tbody>
</table>
Execution Environment
EXECUTION ENVIRONMENT

SET SQL_MODE = '';

DELETE FROM salary WHERE emp_no = 127;

call put(301,10);
ERROR 1264 (22003): Out of range value adjusted for column 'emp_no' at row 1

select emp_no, work_done, bonus from salary;

<table>
<thead>
<tr>
<th>emp_no</th>
<th>work_done</th>
<th>bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>2500</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>2500</td>
</tr>
</tbody>
</table>
Agenda

- Overview
- Stored routines language
- Execution environment
- Stored routines and INFORMATION_SCHEMA
- Using dynamic SQL
- Event scheduler
- Stored routines compared to client side languages
- Resources
- Q&A
INFORMATION SCHEMA = METADATA!
INFORMATION SCHEMA
INFORMATION SCHEMA

• metadata about (among other things)
INFORMATION SCHEMA

- metadata about (among other things)
- tables attributes
INFORMATION SCHEMA

- metadata about (among other things)
  - tables attributes
  - columns
INFORMATION SCHEMA

- metadata about (among other things)
  - tables attributes
  - columns
  - relationships
INFORMATION SCHEMA

- metadata about (among other things)
  - tables attributes
  - columns
  - relationships
  - users
INFORMATION SCHEMA

- metadata about (among other things)
  - tables attributes
  - columns
  - relationships
  - users
  - views
INFORMATION SCHEMA

• metadata about (among other things)
  • tables attributes
  • columns
  • relationships
  • users
  • views
  • routines
INFORMATION SCHEMA
INFORMATION SCHEMA

# manually
SHOW TABLE STATUS LIKE 'table_name'\G

# look at the information
INFORMATION SCHEMA

# manually
SHOW TABLE STATUS LIKE 'table_name'\G

# look at the information

# within a stored routine
SELECT ENGINE, table_rows
FROM information_schema.tables
WHERE table_name = 'table_name'
   AND table_schema = SCHEMA()\G

# use the information
Agenda

• Overview
• Stored routines language
• Execution environment
• Stored routines and INFORMATION_SCHEMA
• **Using dynamic SQL**
• Event scheduler
• Stored routines compared to client side languages
• Resources
• Q&A
Dynamic SQL
Dynamic SQL

Text converted to a query

```sql
set @query = 'select * from salary where emp_no= ? '; 
prepare myStat from @query; 
set @emp = 1; 
EXECUTE myStat USING @emp;
```
Dynamic SQL
Dynamic SQL
example: storing queries on tables

```sql
INSERT INTO QTABLE VALUES ( 1, 'select * from salary where emp_no= ? ');

and later ... 

set @query = (select query from QTABLE where query_id = 1);

prepare myStat from @query;
set @emp = 1;
EXECUTE myStat USING @emp;
```
Why dynamic SQL?
Why dynamic SQL?

• When applying the same action to several objects
Why dynamic SQL?

• When applying the same action to several objects
• To create flexible queries with variable metadata
Why dynamic SQL?

- When applying the same action to several objects
- To create flexible queries with variable metadata
- To overcome SQL limits
Why dynamic SQL?

• When applying the same action to several objects
• To create flexible queries with variable metadata
• To overcome SQL limits
• Especially useful for administration
Dynamic SQL - a complex example
Dynamic SQL - a complex example

getting a list of tables with current row number

# by hand ....
SELECT
    'City' AS t, COUNT(*) as records
FROM world.City
UNION
SELECT
    'Country' AS t, COUNT(*) as records
FROM world.Country
UNION
SELECT
    'CountryLanguage' AS t, COUNT(*) as records
FROM world.CountryLanguage;
Dynamic SQL - a complex example
Dynamic SQL - a complex example

# with a routine ...

SET GROUP_CONCAT_MAX_LEN=10000;
SET @QUERY = CONCAT("SET @Q2 = ( SELECT GROUP_CONCAT(concat("SELECT '",
    TABLE_NAME, "." AS t,
    COUNT(*) as records
    FROM ", table_schema, ".\",
    table_name)
    SEPARATOR ' UNION ' )
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_SCHEMA = ? )");
prepare query1 from @query;
set @db = db_name;
execute query1 using @db;
prepare query2 from @Q2;
execute query2;
Dynamic SQL - a complex example
Dynamic SQL - a complex example

# with a routine ...

call table_list('world');

+-----------------+---------+
| t               | records |
+-----------------+---------+
| City            |  4079   |
| Country         |   239   |
| CountryLanguage |   984   |
+-----------------+---------+

Presented by

MySQL Conference & Expo
Dynamic SQL - a more complex example
Dynamic SQL - a more complex example
# creating a list of CRC values from a table

CREATE FUNCTION column_list(
    p_db_name varchar(50),
    p_table_name varchar(50)
) RETURNS varchar(10000)
BEGIN
    SET GROUP_CONCAT_MAX_LEN=10000;
    RETURN (select GROUP_CONCAT(column_name )
        from INFORMATION_SCHEMA.COLUMNS
        WHERE table_name = p_table_name
        AND table_schema = p_db_name);
END
Dynamic SQL - a more complex example

# creating a list of CRC values from a table

CREATE FUNCTION column_list(
    p_db_name varchar(50),
    p_table_name varchar(50)
)
RETURNS varchar(10000)
BEGIN
    SET GROUP_CONCAT_MAX_LEN=10000;
    RETURN (select GROUP_CONCAT(column_name )
            from
            INFORMATION_SCHEMA.COLUMNS
            WHERE table_name = p_table_name
            AND table_schema = p_db_name);
END

It works. But it will fail on NULL values
Dynamic SQL - a more complex example
Dynamic SQL - a more complex example

# creating a list of CRC values from a table
# using a safer column list

```
SET GROUP_CONCAT_MAX_LEN=10000;
RETURN (select GROUP_CONCAT(
  if(is_nullable = 'yes',
    concat('COALESCE(',
      column_name,', '''), column_name )
  )
FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_name = p_table_name
  AND table_schema = p_db_name);
```
Dynamic SQL - a more complex example
Dynamic SQL - a more complex example

# creating a list of CRC values from a table

CREATE PROCEDURE do_table_row (  
    p_db_name varchar(50),  
    p_table_name varchar(50),  
    func_name varchar(20)
)

BEGIN  
    declare col_list varchar(10000);  
    set col_list = safe_column_list(p_db_name, p_table_name);  
    set @query = CONCAT('SELECT ', func_name, '(CONCAT(', col_list, ')) FROM ', p_db_name, '.', p_table_name );  
    prepare q from @query;  
    execute q;  
END
Dynamic SQL - a more complex example
## Dynamic SQL - a more complex example

# creating a list of CRC values from a table - using the routine

cREATE TABLE t1 (  
id INT NOT NULL,  
dt DATETIME NOT NULL,  
who VARCHAR(20));

INSERT INTO t1 VALUES  
(1, NOW(), null),  
(2, NOW() - INTERVAL 2 HOUR, 'abc');

Query OK, 2 rows affected (0.04 sec)  
Records: 2  Duplicates: 0  Warnings: 0
Dynamic SQL - a more complex example
Dynamic SQL - a more complex example

# creating a list of CRC values from a table - using the routine

call do_table_row(schema(),'t1','md5');
call do_table_row(schema(),'t1','SHA1');
Agenda

• Overview
• Stored routines language
• Execution environment
• Stored routines and INFORMATION_SCHEMA
• Using dynamic SQL
• Event scheduler
• Stored routines compared to client side languages
• Resources
• Q&A
Event scheduler
Event scheduler

- Available in MySQL 5.1
Event scheduler

- Available in MySQL 5.1
- temporal triggers
Event scheduler

- Available in MySQL 5.1
- temporal triggers
- execute "AT" a given time or "EVERY" x time intervals
Event scheduler

- Available in MySQL 5.1
- temporal triggers
- execute "AT" a given time or "EVERY" x time intervals
- associated to a given schema
Event scheduler

- Available in MySQL 5.1
- temporal triggers
- execute "AT" a given time or "EVERY" x time intervals
- associated to a given schema
- run with definer grants
EVENT SCHEDULER syntax
EVENT SCHEDULER syntax

CREATE EVENT event_name
ON SCHEDULE AT NOW() + INTERVAL 1 HOUR
DO
    CALL some_nice_procedure();

CREATE EVENT event_name
ON SCHEDULE EVERY 3 minutes
DO
    CALL check_if_everything_OK();
Event scheduler - when and how
Event scheduler - when and how

• data warehouse import (with federated tables)
Event scheduler - when and how

• data warehouse import (with federated tables)
• data consolidation
Event scheduler - when and how

- data warehouse import (with federated tables)
- data consolidation
- sanity checks
Event scheduler - when and how

• data warehouse import (with federated tables)
• data consolidation
• sanity checks
  • user profiles
Event scheduler - when and how

- data warehouse import (with federated tables)
- data consolidation
- sanity checks
  - user profiles
  - tables
Event scheduler - when and how

• data warehouse import (with federated tables)
• data consolidation
• sanity checks
  • user profiles
  • tables
• creation of time-dependent objects
EVENT SCHEDULER in depth
EVENT SCHEDULER in depth

Using Triggers and Events for MySQL Administration and Auditing
EVENT SCHEDULER in depth

Using Triggers and Events for MySQL Administration and Auditing

Tobias Asplund, MySQL AB
EVENT SCHEDULER in depth

Using Triggers and Events for MySQL Administration and Auditing

Tobias Asplund, MySQL AB

Date: Thursday, April 26
EVENT SCHEDULER in depth

Using Triggers and Events for MySQL Administration and Auditing

Tobias Asplund, MySQL AB

Date: Thursday, April 26
Time: 2:30pm - 3:15pm
EVENT SCHEDULER in depth

Using Triggers and Events for MySQL Administration and Auditing

Tobias Asplund, MySQL AB

Date: Thursday, April 26
Time: 2:30pm - 3:15pm
Location: Ballroom B
Agenda

• Overview
• Stored routines language
• Execution environment
• Stored routines and INFORMATION_SCHEMA
• Using dynamic SQL
• Event scheduler
• Stored routines compared to client side languages
• Resources
• Q&A
## Performance Overview

<table>
<thead>
<tr>
<th>task</th>
<th>standard SQL</th>
<th>Stored routines</th>
<th>external programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>finding prime numbers</td>
<td>N/A</td>
<td>extremely slow</td>
<td>fast</td>
</tr>
<tr>
<td>fetching and modifying records</td>
<td>extremely fast (*)</td>
<td>fast</td>
<td>slow</td>
</tr>
<tr>
<td>self joining a table</td>
<td>slow</td>
<td>fast</td>
<td>extremely slow</td>
</tr>
<tr>
<td>update correlated query</td>
<td>slow</td>
<td>fast</td>
<td>slow</td>
</tr>
</tbody>
</table>

(*) when applicable
Number crunching (don't do that)

(*) from the book *MySQL Stored Procedure Programming*
Fetching records across the network

(*) from the book *MySQL Stored Procedure Programming*
Optimizing a complex self join

(*) from the book *MySQL Stored Procedure Programming*
Optimizing a correlated nested query

(*) from the book MySQL Stored Procedure Programming
Agenda

• Overview
• Stored routines language
• Execution environment
• Stored routines and INFORMATION_SCHEMA
• Using dynamic SQL
• Event scheduler
• Stored routines compared to client side languages
• **Resources**
• Q&A
Resources

- MySQL manual

http://dev.mysql.com/doc

Chapter 17. Stored Procedures and Functions

Table of Contents

17.1. Stored Routines and the Grant Tables
17.2. Stored Routine Syntax
  17.2.1. CREATE PROCEDURE and CREATE FUNCTION Syntax
  17.2.2. ALTER PROCEDURE and ALTER FUNCTION Syntax
  17.2.3. DROP PROCEDURE and DROP FUNCTION Syntax
  17.2.4. CALL Statement Syntax
  17.2.5. BEGIN ... END Compound Statement Syntax
  17.2.6. DECLARE Statement Syntax
  17.2.7. Variables in Stored Routines
  17.2.8. Conditions and Handlers
  17.2.9. Cursors
  17.2.10. Flow Control Constructs

17.3. Stored Procedures, Functions, Triggers, and LAST_INSERT_ID
17.4. Binary Logging of Stored Routines and Triggers

Stored routines (procedures and functions) are supported in MySQL 5.0. A stored procedure is a set of SQL statements that can be stored in the server. Once this has been done, clients don’t need to keep reissuing the individual statements but can refer to the stored procedure instead.

Answers to some questions that are commonly asked regarding stored routines in MySQL can be found in Section A.4, “MySQL 5.0 FAQ — Stored Procedures”.
Resources

• MySQL stored procedure programming

http://www.oreilly.com/catalog/mysqlspp
Resources

• MySQL stored routines library (http://forge.mysql.com)

• global variables

• simple and complex data structures

• loops

• syntax helpers

• named parameters

• test units
Resources

• blogs

Planet MySQL
I was out at lunch with some clients and started discussing properties of database systems and the idea of MySQL not having stored procedures popped up its ...
www.planetmysql.org/entries/6756 - 5k - Cached - Similar pages - Note this

Planet MySQL
MySQL stored procedures can use SELECT statements in their body, returning a data set to the caller. The manual also explains that multiple data sets can be ...
www.planetmysql.org/entries/6780 - 6k - Cached - Similar pages - Note this

Planet MySQL
This is a tale about Ruby On Rails, custom stored procedures for MySql 5 and how ... The brave adventurer may try to create a Mysql 5 stored procedures like ...
www.planetmysql.org/entries/6682 - 10k - Cached - Similar pages - Note this

Planet MySQL
MySQL stored procedures can use SELECT statements in their body, ... See the complete example at Multiple data sets in MySQL stored procedures ...
www.planetmysql.org/?offset=10 - 38k - Cached - Similar pages - Note this
Resources

• MySQL Forge

Resources

When **NOT** to use stored routines

- SQL Cookbook
- The Art of SQL
- MySQL Cookbook

---

Presented by

MySQL Conference & Expo

O'REILLY®
Agenda

- Overview
- Stored routines language
- Execution environment
- Stored routines and INFORMATION_SCHEMA
- Using dynamic SQL
- Event scheduler
- Stored routines compared to client side languages
- Resources
- Q&A
Questions & Answers

Any questions?
Thank you!

- http://dev.mysql.com/
- http://www.mysqlconf.com/
- giuseppe@mysql.com