MySQL - RDBMS

Trainer: Mr. Nilesh Ghule

Constraints

- Constraints are restrictions imposed on columns.
- There are five constraints
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK
- Few constraints can be applied at either column level or table level. Few constraints can be applied on both.
- Optionally constraint names can be mentioned while creating the constraint. If not given, it is auto-generated.
- Each DML operation check the constraints before manipulating the values. If any constraint is violated, error is raised.

Constraints

- PRIMARY KEY
  - Column or set of columns that uniquely identifies a row.
  - Only one primary key is allowed for a table.
  - Primary key column cannot have duplicate or NULL values.
  - Internally index is created on PK column.
  - TEXT/BLOB cannot be primary key.
  - If no obvious choice available for PK, composite or surrogate PK can be created.
  - Creating PK for a table is a good practice.
  - PK can be created at table level or column level.
  - CREATE TABLE table(c1 TYPE PRIMARY KEY, …);
  - CREATE TABLE table(c1 TYPE, …, PRIMARY KEY(c1));
  - CREATE TABLE table(c1 TYPE, …, CONSTRAINT constraint_name PRIMARY KEY(c1));
  - CREATE TABLE table(c1 TYPE, c2 TYPE, …, PRIMARY KEY(c1, c2));

Constraints

- FOREIGN KEY
  - Column or set of columns that references a column of some table.
  - If column belongs to the same table, it is "self referencing".
  - Foreign key constraint is specified on child table column.
  - FK can have duplicate values as well as null values.
  - FK constraint is applied on column of child table (not on parent table).
  - Child rows cannot be deleted, until parent rows are deleted.
  - MySQL have ON DELETE CASCADE clause to ensure that child rows are automatically deleted, when parent row is deleted. ON UPDATE CASCADE clause does same for UPDATE operation.
  - By default foreign key checks are enabled. They can be disabled by
    - SET @@foreign_key_checks = 0;
  - FK constraint can be applied on table level as well as column level.
  - CREATE TABLE child(c1 TYPE, …, FOREIGN KEY (c1) REFERENCES parent(col))

Constraints

- CHECK
  - CHECK is integrity constraint in SQL.
  - CHECK constraint specifies condition on column.
  - Data can be inserted/updated only if condition is true; otherwise error is raised.
  - CHECK constraint can be applied at table level or column level.
  - CREATE TABLE table(c1 TYPE, c2 TYPE CHECK condition1, …, CHECK condition2);

DDL – ALTER statement

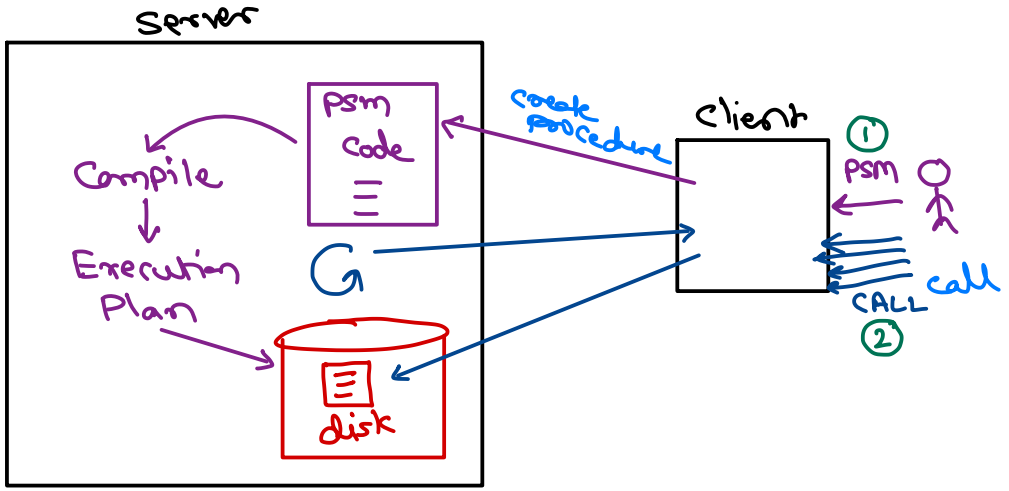
- ALTER statement is used to do modification into table, view, function, procedure, …
- ALTER TABLE is used to change table structure.
- Add new column(s) into the table.
  - ALTER TABLE table ADD col TYPE;
  - ALTER TABLE table ADD c1 TYPE, c2 TYPE;
- Modify column of the table.
  - ALTER TABLE table MODIFY col NEW_TYPE;
- Rename column.
  - ALTER TABLE CHANGE old_col new_col TYPE;
- Drop a column
  - ALTER TABLE table name DROP COLUMN col;
- Rename table
  - ALTER TABLE table RENAME TO new_table;

MySQL Programming

- RDBMS Programming is an ISO standard – part of SQL standard – since 1992.
- SQL/PSM stands for Persistent Stored Module.
- Inspired from PL/SQL - Programming language of Oracle.
- PSM allows writing programs for RDBMS. The program contains set of SQL statements along with programming constructs e.g. variables, if-else, loops, case, ...
- PSM is a block language. Blocks can be nested into another block.
- MySQL program can be a stored procedure, function or trigger.

MySQL Programming

- MySQL PSM program is written by db user (programmers).
- It is submitted from client, server check syntax & store them into db in compiled form.
- The program can be executed by db user when needed.
- Since programs are stored on server in compiled form, their execution is very fast.
- All these programs will run in server memory.



Stored Procedure

- Stored Procedure is a routine. It contains multiple SQL statements along with programming constructs.
- Procedure doesn't return any value (like void fns in C).
- Procedures can take zero or more parameters.
- Procedures are created using CREATE PROCEDURE and deleted using DROP PROCEDURE.
- Procedures are invoked/called using CALL statement.
- Result of stored procedure can be
  - returned via OUT parameter.
  - inserted into another table.
  - produced using SELECT statement (at end of SP).
- Delimiter should be set before writing procedure.

Stored Procedure

```
CREATE TABLE result(v1 DOUBLE, v2 VARCHAR(50));

DELIMITER $$

CREATE PROCEDURE sp_hello()
BEGIN
    INSERT INTO result VALUES(1, 'Hello World');
END;
$$

DELIMITER ;

CALL sp_hello();

SELECT * FROM result;
```

```
-- 01_hello.sql (using editor)
DROP PROCEDURE IF EXISTS sp_hello;
DELIMITER $$
CREATE PROCEDURE sp_hello()
BEGIN
    SELECT 1 AS v1, 'Hello World' AS v2;
END;
$$
DELIMITER ;


SOURCE /path/to/01_hello.sql

CALL sp_hello();
```

Stored Procedure – PSM Syntax

VARIABLES

```
DECLARE varname DATATYPE;
DECLARE varname DATATYPE DEFAULT init_value;
SET varname = new_value;
SELECT new_value INTO varname;
SELECT expr_or_col INTO varname FROM table_name;
```

PARAMETERS

```
CREATE PROCEDURE sp_name(PARAMTYPE p1 DATATYPE)
BEGIN
    …
END;

-- IN param: Initialized by calling program.
-- OUT param: Initialized by called procedure.
-- INOUT param: Initialized by calling program and
modified by called procedure
-- OUT & INOUT param declared as session variables.

CREATE PROCEDURE sp_name(OUT p1 INT)
BEGIN
    SELECT 1 INTO p1;
END;

SET @res = 0;
CALL sp_name(@res);
SELECT @res;
```

IF-ELSE

```
IF condition THEN
     body;
END IF;
-----------------------
IF condition THEN
     if-body;
ELSE
     else-body;
END IF;
-----------------------
IF condition THEN
     if1-body;
ELSE
     IF condition THEN
            if2-body;
     ELSE
            else2-body;
     END IF;
END IF;
-----------------------
IF condition THEN
     if1-body;
ELSEIF condition THEN
     if2-body;
ELSE
     else-body;
END IF;
```

LOOPS

```
WHILE condition DO
    body;
END WHILE;
------------------
REPEAT
    body;
UNTIL condition
END REPEAT;
------------------
label: LOOP
IF condition THEN
        ...
        LEAVE label;
    END IF;
    ...
END LOOP;
```

CASE-WHEN

```
CASE
WHEN condition THEN
      body;
WHEN condition THEN
      body;
ELSE
       body;
END CASE;
```

SHOW PROCEDURE

```
SHOW PROCEDURE STATUS
LIKE ‘sp_name’;

SHOW CREATE PROCEDURE sp_name;
```

DROP PROCEDURE

```
DROP PROCEDURE
IF EXISTS sp_name;
```

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>