Databases

PRAGMATIC IT Learning & Outsourcing Center

Other Database Objects

www.pragmatic.bg

Lector: Hristo Topuzov
E-mail: Hristo.Topuzov@pragmatic.bg

2018

Agenda

- Views
- Triggers
- Indexes
- Store Procedures

Views

- View is a data object which does not contain any data
- Contents of the view are the resultant of a SELECT queries. If data is changed in the underlying table, the same change is reflected in the view
- They are operated just like base table but they don't contain any data of their own
- Views simplify queries
- Views can be used to imply security on the data they represent

Views

- Following statements create a view:

```
CREATE VIEW view_name
AS select_statement
```

- Example:

```
CREATE VIEW V_MARKETING_EMPLOYEES
AS
SELECT E.NAME, D.NAME DEPARTMENT_NAME
FROM EMPLOYEES E
JOIN DEPARTMENTS D ON D.ID = E.DEPARTMENTID
WHERE D.NAME = 'Marketing'
```

Views

- Views are roughly divided in two types - simple and complex
- Simple views:
  - SELECT from one table
  - no SQL functions and GROUP BY clauses
  - DML statements can typically be used (INSERT,UPDATEandDELETE)
- Complex views:
  - SELECT from one or more tables
  - can use SQL functions and GROUP BY clauses
  - DML statements typically cannot be used (INSERT,UPDATE And DELETE)

Views

- ALTER VIEW statement changes the definition of an existing view

```
ALTER  VIEW view_name
AS select_statement
```

- DROP VIEW statement is used to remove one or more views

```
DROP VIEW view_name
```

Triggers

- A trigger is a set of actions that are run automatically when a specified change operation (SQL INSERT, UPDATE, or DELETE statement) is performed on a specified table
- Triggers are attached to a specific table
- In MySQL triggers may be executed at the following points in time:
  - before a row is added/deleted/modified
  - after a row is added/deleted/modified

Triggers

- Triggers are typically used for:
  - Logging information about data changes to the tables
  - Archiving data
  - Rejecting table manipulations if some criteria is not met
  - Checking data before/after manipulations
  - Showing users a message when a command is executed

Triggers

- Triggers are typically used for:
  - Global enforcement of business rules. Define a trigger once and then reuse it for any application that uses the database
  - Easier maintenance. If a business policy changes, you need to change only the corresponding trigger program instead of each application program.

Triggers

- Triggers are created with the CREATE TRIGGER command

```
CREATE
TRIGGER trigger_name
trigger_time trigger_event
ON tbl_name FOR EACH ROW
trigger_body
trigger_time: { BEFORE | AFTER }
trigger_event: { INSERT | UPDATE | DELETE }
```

Triggers

- Example:

```
CREATE TRIGGER Test AFTER UPDATE on Vendors
FOR EACH ROW
BEGIN
insert into messages
select concat ('trigger3 executed', old.id, new.id);
```

Indexes

- Indexes are special lookup tables that are used to find rows with specific column values **quickly**.
- Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows
- Indexes are created on one or more columns
- Indexes **slow down** DML queries (INSERT,UPDATE and DELETE) since the index must be rebuilt

Indexes

- Indexes can be unique (meaning all values in the indexed columns must be unique) or non-unique
- Indexes are automatically created from PRIMARY and UNIQUE key constraints

Indexes

- Indexes are created typically on columns that:
  - are primary/foreign keys that participate often in JOIN queries
  - are used often in queries that retrieve values based on a range (e.g. values between two dates)
  - participate often in sorting operations in queries (in an ORDER BY clause)
  - participate often in aggregation queries (in a GROUP BY clause)

Indexes

- Indexes are typically not created on columns that:
  - have a small number of unique values
  - are rarely used in queries
- Here is the syntax to create an Index on a table
  - CREATE UNIQUE INDEX index_name ON table_name (column1, column2,...);
- DROP INDEX Syntax
  - DROP INDEX index_name ON tbl_name

Stored Procedures

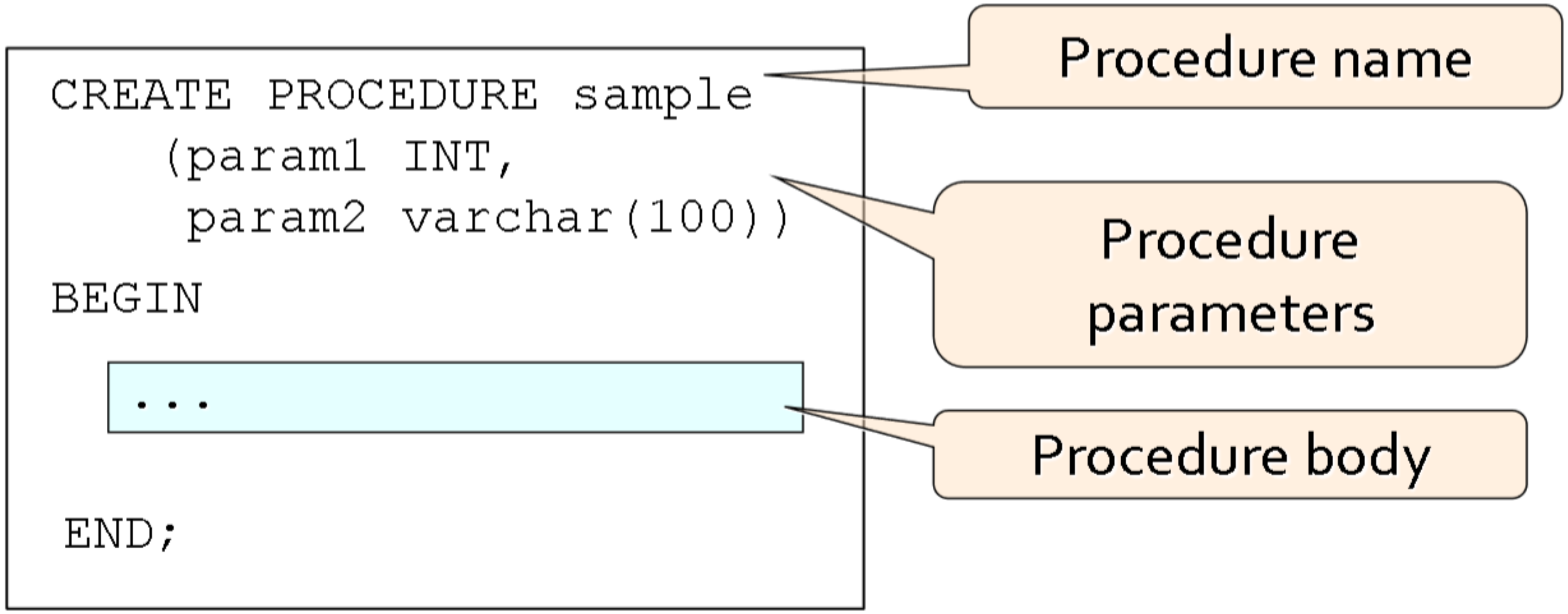
- A stored procedure is a set of SQL statements that can be stored in the server.
- Stored procedures are named procedures that can be executed repeatedly on the database server
- Stored procedures can take parameters
- Stored procedures combine SQL statements and programming logic

Stored Procedures

- Stored procedures are secure. Database administrator can grant appropriate permissions to applications that access stored procedures in the database without giving any permission on the underlying database tables
- Stored procedures helps reduce the traffic between application and database server
- Stored procedures are reusable

Stored Procedures

- Stored procedures are program logic and have the following structure:



```
CREATE PROCEDURE sample
      (param1 INT,
       param2 varchar(100))
BEGIN
   ...

END;
```

Stored Procedures

- Once defined routines can be called with the CALL command
- For example:

```
CALL sample
```

- A stored routine can also be called from another stored routine

Stored Procedures

- Parameters of the stored procedure can additionally be defined as:
- IN -parameter has initial value when passed to the procedure but is not modified after procedure finishes
- OUT -parameter might be modified by the procedure and used with modified value after procedure finishes
- INOUT -parameter has an initial value and might be modified by the procedure

Stored Procedures

- Example:

```
delimiter //
CREATE  PROCEDURE sample( IN param1 INT, OUT
param2 INT, INOUT param3 INT)
BEGIN
set param1 = 1;
set param2 = param3;
set param3 = 3;
END//
```

Stored Procedures

- Example:

```
set @var1 = 0;
set @var2 = 0;
set @var3 = 0;
CALL sample(@var1, @var2, @var3);
select @var1;
select @var2;
select @var3;
```

Questions