Database objects and metadata

Agenda

- Creating and Manipulating Triggers
- Creating and Manipulating Indexes
- Creating and Manipulating Events
- Database Metadata

Creating and Manipulating Triggers

Creating and Manipulating Triggers

- Triggers are attached to a specific table
- They are executed when a certain condition occurs (such as an `INSERT`, `UPDATE` or `DELETE` operation)

Creating and Manipulating 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

Creating and Manipulating Triggers

- 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

Creating and Manipulating Triggers

- Triggers are created with the `CREATE TRIGGER` command
- General syntax:

```
CREATE
[DEFINER = { user | CURRENT_USER }]
TRIGGER [trigger_name]
[trigger_time] [trigger_event]
ON tbl_name FOR EACH ROW
trigger_body
```

Creating and Manipulating Triggers

Example:

```
delimiter \\
create trigger trigger3 after update on Vendors
FOR EACH ROW
begin
  insert into messages(msg) select
concat('trigger3 executed', old.id, new.id);
end \\
```

Creating and Manipulating Indexes

Creating and Manipulating Indexes

- Indexes are used to improve the performance of certain types of `SELECT` queries
- Indexes are created on one or more columns
- Indexes are implemented by means of special data structure such as a B-tree or a bitmap based on the type of index

Creating and Manipulating Indexes

- Indexes typically have a memory footprint when created
- Indexes slow down DML queries (`INSERT`, `UPDATE` and `DELETE`) since the index must be rebuilt

Creating and Manipulating 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

Creating and Manipulating 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)

Creating and Manipulating Indexes

- Indexes are created typically on columns that:
  - participate often in sorting operations in queries (in an `ORDER BY` clause)
  - participate often in aggregation queries (in a `GROUP BY` clause)

Creating and Manipulating Indexes

- Indexes are typically not created on columns that:
  - have a small number of unique values
  - are rarely used in queries

Creating and Manipulating Indexes

- Types of indexes in MySQL database:
  - B-tree index
  - Bitmap index

Creating and Manipulating Indexes

- B-tree index - the standard type of index in a MySQL database - useful when selecting values in a range and is created with the `CREATE INDEX` command

- Example:

```
create index salary_ind on Employees(Salary);
```

Creating and Manipulating Indexes

- Bitmap index - for columns with a small number of unique values and is typically used when data is loaded in chunks

- Example:

```
create index status_ind using hash on Vacations(Status);
```

Creating and Manipulating Indexes

- An index can be dropped with the DROP INDEX command

- Example:

```
drop index salary_ind;
```

Creating and Manipulating Events

Creating and Manipulating Events

- Relational database systems typically provide mechanisms for scheduled execution of tasks
- MySQL events are tasks that run according to a schedule
- Oracle database provides the DBMS_JOB PL/SQL package for creating scheduled jobs

(note: you can also schedule tasks in your OS - e.g. crontab for Unix and Windows Event Scheduler for Windows)

Creating and Manipulating Events

- In order to create an event (scheduled job) in MySQL the **event_scheduler** thread must be enabled:

```
SET event_scheduler=on;
```

- In order to check that the **event_scheduler** thread is running you can display all current MySQL processes using:

```
SHOW PROCESSLIST
```

Creating and Manipulating Events

- After the MySQL scheduler process is enabled you can schedule jobs by creating events (an event is created in the current database)

- There are two types of events:
    - one time events - executed only once
    - repeating events - executed multiple times

Creating and Manipulating Events

- General syntax for creating an event:

```
CREATE
[DEFINER = { user | CURRENT_USER }]
EVENT
[IF NOT EXISTS]
event_name
ON SCHEDULE schedule
[ON COMPLETION [NOT] PRESERVE]
[ENABLE | DISABLE | DISABLE ON SLAVE]
[COMMENT 'comment']
DO event_body;
```

Creating and Manipulating Events

- General syntax for a schedule in the `ON SCHEDULE` clause:

```
AT timestamp [+ INTERVAL interval] ...
| EVERY interval
[STARTS timestamp [+ INTERVAL interval] ...]
[ENDS timestamp [+ INTERVAL interval] ...]
```

Creating and Manipulating Events

- Example (one time event):

```
CREATE EVENT one_time_event
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
DO
UPDATE hrm.event_counter SET counter = counter + 1;
```

Creating and Manipulating Events

- Example (repeating event):

```
CREATE EVENT repeating_event
ON SCHEDULE EVERY 1 MINUTE
DO
UPDATE hrm.event_counter SET counter = counter + 1;
```

Creating and Manipulating Events

- Events can be changed using the `ALTER EVENT` command without having to delete them

- Events can be deleted using the `DROP EVENT` command

Database Metadata

Database Metadata

- Database metadata refers to information about the database objects
- MySQL provides various utilities to retrieve database metadata such as MySQL functions and the **information_schema** database

Database Metadata

- Databases in a MySQL server can be retrieved using the `SHOW DATABASES` command
- To switch to a particular database you can use the `USE DATABASE <name>` command
- To delete a database you can use the `DROP DATABASE <name>` command

Database Metadata

- To check the current database that is being used you can use the `database()` function:

```
SELECT DATABASE()
```

- To view database tables in the current database you can use the `SHOW TABLES` command

Database Metadata

- To check the currently logged user in the database you can issue:

```
SELECT USER( )
```

Database Metadata

- To check server status indicators for the current session you can issue:

```
SHOW STATUS
```

- To check server status indicators for all sessions you can issue:

```
SHOW GLOBAL STATUS
```

Database Metadata

- For example the following retrieves the number of `SELECT` queries issued in the current session:

```
SHOW STATUS where variable_name = 'Com_select';
```

- The MySQL reference guide provides information about the status indicators listed from `SHOW STATUS`

Questions ?