CSIT115 Data Management and Security

Architecture of Relational Database Server

Dr Janusz R. Getta

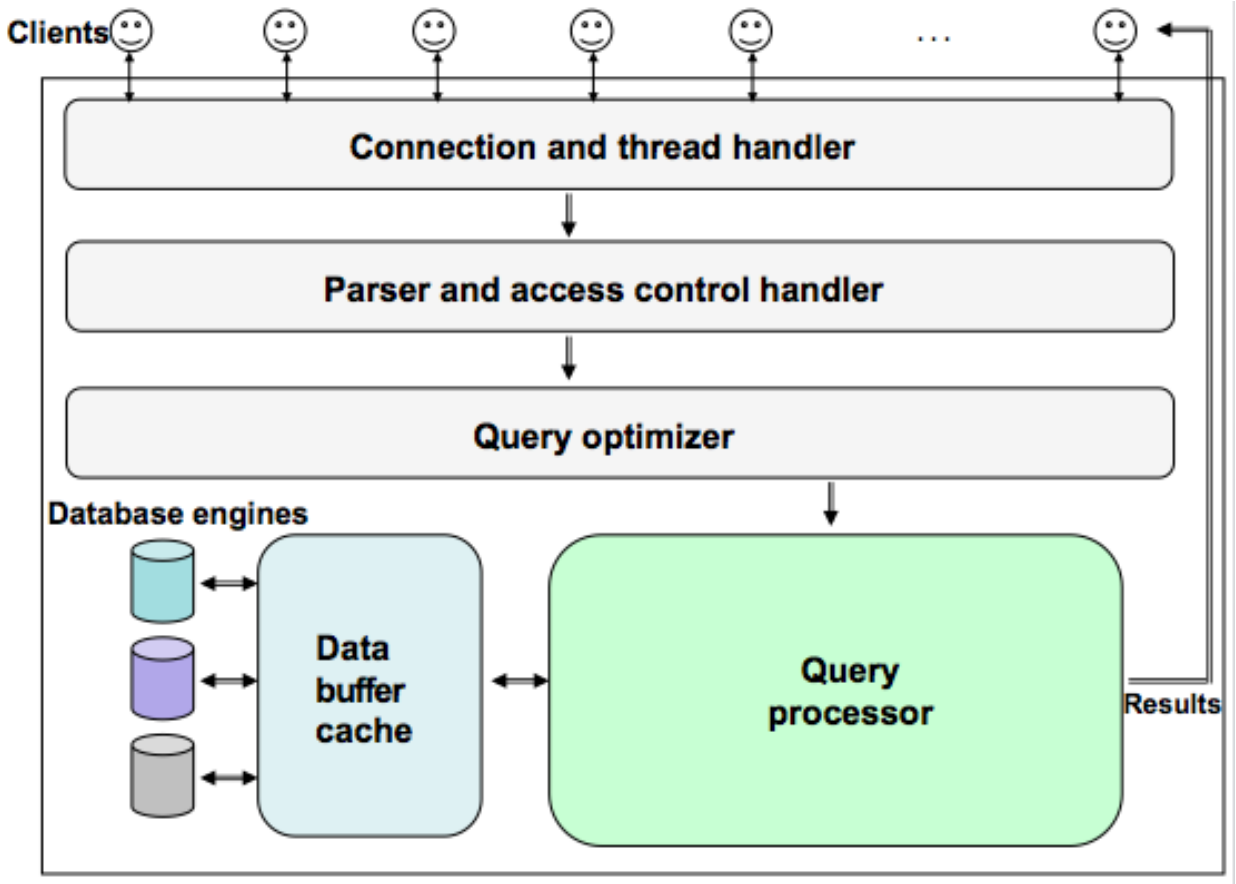
School of Computing and Information Technology -
University of Wollongong

Architecture of Relational Database Server

Outline

Client-Server Architecture

A client-server architecture means a number of clients connected to a database server over a local or wide-area network



Client-Server Architecture

A database server is implemented over there layers:

- Services not unique to MySQL like: network based client/server tools for connection handling, authentication, security, etc
- DDL and DML processing like query processing, analysis, optimization, caching, and all built-in functions, data entry, data modification, creating database structures
- Database (storage) engines responsible for storing and retrieving data, e.g. InnoDB, MyISAM, MEMORY, CSV, etc

Client-Server Architecture

Client connections are organized in the following way:

- As more than one client can be connected at a time each client connection gets its own thread within a database server
- A thread resides on one core or on one CPU
- When a client connects to a server then a server authenticates a connection
- Authentication is based on a user name, originating host, and password
- Once a client is connected, the server verifies whether it has the privileges to access the relational tables in a database

Architecture of Relational Database Server

Outline

Basic Operations on Database Server

Starting/stopping a database server

Usually a database server is automatically started at boot-up time of operation system

A database server can be stopped from a command line for a certain period of time

Stopping MySQL server

```
service mysql stop
```

A database server can be started from a command line

Starting MySQL server

```
service mysql start
```

A status of database server can be found in the following way

Listing a status of MySQL server

```
service mysql status
```

Architecture of Relational Database Server

Outline

Initialization Variables

At startup time a database server reads the system initialization variables

The system initialization variables determine functionality of a database server

For example, some of system initalization variable of MySQL:

System initialization variables

```
+------------------------------+--------------------------+
| Variable_name                | Value                    |
+------------------------------+--------------------------+
| auto_increment_increment     | 1                        |
| auto_increment_offset        | 1                        |
| autocommit                   | ON                       |
| automatic_sp_privileges      | ON                       |
| avoid_temporal_upgrade       |                          |
...          ...               ...          ...          ...
| datadir                      | /var/lib/mysql/          |
| date_format                  | %Y-%m-%d                 |
| lower_case_table_names       | 0                        |
| max_user_connections         | 0                        |
...          ...               ...          ...          ...
+------------------------------+--------------------------+
```

Initialization Variables

System initialization variables are included in system configuration files

To find locations of system configuration files execute the following commands at command prompt

Listing location of system configuration files

```
mysql --help  | grep "Default options" -A 1
```

Typically the configuration files are located at

Location of system configuration files

```
/etc/my.cnf
/etc/mysql/my.cnf
~/.my.cnf
```

In our case a file with system initialization variables `mysqld.cnf` is located at

Location of system configuration files

```
/etc/mysql/mysql.conf.d/
```

Initialization Variables

The sample contents of system configuration file can be the following

System initialization variables

```
[mysqld]

user   = mysql
pid-file = /var/run/mysqld/mysqld.pid
socket   = /var/run/mysqld/mysqld.sock
port   = 3306
basedir    = /usr
datadir    = /var/lib/mysql
tmpdir   = /tmp
lc-messages-dir  = /usr/share/mysql
explicit_defaults_for_timestamp
```

Initialization Variables

To display ALL (501) system initialization variables use `show variables` statement

To display GLOBAL (487) system initialization variables use `show global variables` statement (parameters for new connections)

To display LOCAL(SESSION) (487) system initialization parameters use `show variables` statement (parameters for the current connection)

For example, to find all variables related to updates we use a statement

```
Listing system initialization variables with 'update' in their names
show variables like '%update%';
```

For example, to find a value of variable `lower_case_table_names` we use a statement

```
Listing system initialization variables whose name starts from 'lower_case_table'
show variables like 'lower_case_table%'
```

Initialization Variables

To change a value of dynamic system initialization variables we use `set` statement

For example to change a value of system initialization variable `sql_safe_updates` to `0` we use the following statement

Changing a value of system initialization variable

```
set sql_safe_updates=0
```

Some of the system initialization variables are not dynamic and it cannot be changed with `set` !

For example, a variable `lower_case_table_names` is not dynamic and it cannot be changed with `set`

Changing a value of system initialization variable

```
set lower_case_table_names=1
```

Error message

```
ERROR 1238 (HY000): Variable 'lower_case_table_names' is a read only variable
```

The system initialization variables that or not dynamic must be changed in a system configuration file (stop server, change variable, start server)

Architecture of Relational Database Server

Outline

Post Installation

Just after installation there is only one user `root` available on the installed system

First we start `mysql` client and we connect as a user `root` without a password !

The first start of mysql client

```
mysql -u root
```

As it is an evident security risk you must set a password for `root` user

Changing password

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'password';
```

To find what other users can connect to the system execute a statement:

Finding all users that can connect to the system

```
SELECT user, host, HEX(authentication_string) FROM mysql.user;
```

Sample listing of user names

```
+-----------+-----------+-------------------------------------------...------+
| user      | host      | HEX(authentication_string)                        |
+-----------+-----------+-------------------------------------------...------+
| root      | localhost | 2A3831463545323145333534303744384386426432...30394 |
| mysql.sys | localhost | 2A5448495349534E4F544156414C4944504155345...44484 |
| csit115   | localhost | 2A32304534394632443230333737373939383133336...44433 |
+-----------+-----------+-------------------------------------------...------+
```

Post Installation

After a password to a user `root` is changed any future connection as a user `root` must be done as follows

```
mysql -u root -p -v
```

Sample connection as a user root

Finally, remember that MySQL `root` user is completely different from operating system `root` user!

To find what database are available on the system use a statement:

```
show databases;
```

Listing databases

A database `information_schema` is commonly called as a data dictionary and it contains information about relational tables, columns, constraints, etc

To list the names included in a database `information_schema` we switch to the database

```
use information_schema;
```

Switching to 'infomation_schema' database

Post Installation

And then we simply "say"

Listing table names

```
show tables;
```

Now it is possible to access data dictionary tables

For example we access a table `user_privileges` to find what privileges are granted to the users

Listing user privileges

```
SELECT * FROM user_privileges;
```

Post Installation

The privileges granted to the users `root` and csit115 are the following

User privileges

```
+------------------------+---------------+----------------+--------------+
| GRANTEE                | TABLE_CATALOG | PRIVILEGE_TYPE | IS_GRANTABLE |
+------------------------+---------------+----------------+--------------+
| 'root'@'localhost'     | def           | SELECT         | YES          |
| 'root'@'localhost'     | def           | INSERT         | YES          |
| 'root'@'localhost'     | def           | UPDATE         | YES          |
| 'root'@'localhost'     | def           | DELETE         | YES          |
| 'root'@'localhost'     | def           | CREATE         | YES          |
| 'root'@'localhost'     | def           | DROP           | YES          |
...                      ...             ...        ...              ...
| 'csit115'@'localhost'  | def           | SELECT         | NO           |
| 'csit115'@'localhost'  | def           | INSERT         | NO           |
| 'csit115'@'localhost'  | def           | UPDATE         | NO           |
| 'csit115'@'localhost'  | def           | DELETE         | NO           |
...                      ...             ...        ...              ...
+------------------------+---------------+----------------+--------------+
```

Architecture of Relational Database Server

Databases

In order to create the relational tables we have to create a database first

While connected as a user `root` execute a statement

Creating a database

```
CREATE DATABASE database-name;
```

It is simple to drop it with

Dropping a database

```
DROP DATABASE database-name;
```

To be able to create a database a user must have `CREATE` privilege and to drop a database a user must have `DROP` privilege

To list all created database we execute a statement

Listing available databases

```
show databases;
```

To create a relational table in a given database we execute a statement

Setting a default database

```
use database-name;
```

Databases

For example, to use a database `csit115` we execute a statement

Setting 'csit115' database as a default database

```
use csit115;
```

Then, if a relational table `DEPARTMENT` is created in a database `csit115` then we can access the table with a simple

SELECT statement

```
SELECT * FROM DEPARTMENT;
```

Then, if a relational table `COURSE` is created in a database `university` then we can access it either through

Setting 'university' database as a default database

```
use university;
SELECT * FROM COURSE;
```

or through prefixing a table name with a database name

SELECT statement

```
SELECT * FROM university.COURSE;
```

Databases

A user can be connected to through application of `use` statement to one database at a time

A user can access many databases at a time through prefixing the names of relational tables in other databases with an appropriate `database name`

A database can be dropped with

Dropping a database

```
DROP DATABASE database-name;
```

References

Cabral S., Murphy K., MySQL Administrator's Bible, Wiley Publications, 2009 (Available online through UOW Library)

How to ... ? Cookbook, How to manage MySQL database server ? Recipes 8.1, 8.2, 8.3, and 8.4