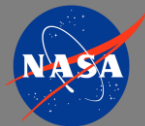


# MySQL Database Administrator

Nobelprog

# MySQL Server Usage



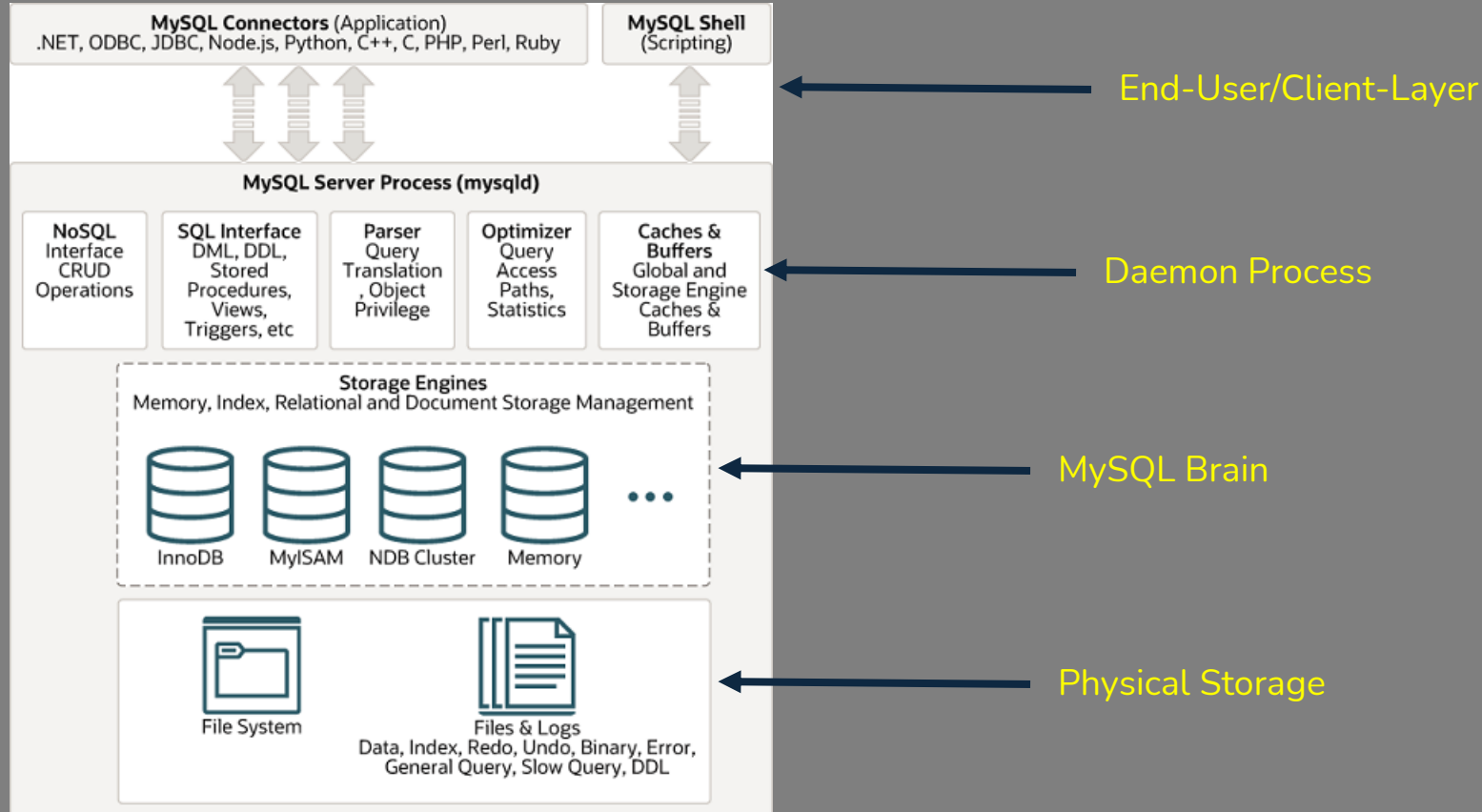
# Course Introduction

- ❖ New Database Installations, Customization, Backup & Recovery
- ❖ Database Management -
- ❖ Database Upgrades - Both Major and Minor Upgrades
- ❖ InnoDB Storage Engine Tuning
- ❖ Database Replication, Troubleshooting, Performance Tuning

# Course Objectives

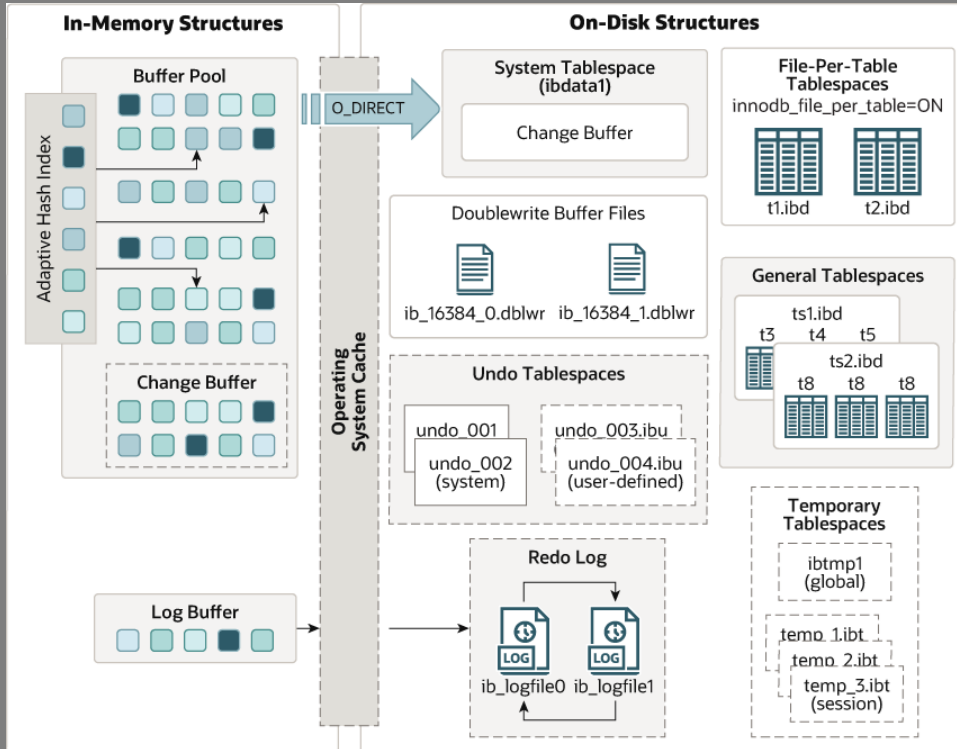
- ❖ MySQL Server Installation
- ❖ Exploring MySQL Server
- ❖ MySQL Server Database Administration
- ❖ MySQL Storage Engines
- ❖ MySQL User Administration
- ❖ MySQL Server Configuration
- ❖ InnoDB Storage Engine
- ❖ MySQL Backup & Restore
- ❖ MySQL Replication
- ❖ Upgrading MySQL Server
- ❖ MySQL Performance and Monitoring

# MySQL Architecture



# MySQL InnoDB Architecture

## InnoDB Architecture:



### In-Memory Structure:

- ❖ **Buffer Pool** - Area in main memory where InnoDB caches table & index data as it is accessed
- ❖ **Change Buffer** - caches changes to non-clustered indexes
- ❖ **Adaptive Hash Index** - acts like in-memory db
- ❖ **Log Buffer** - memory area that holds data to be written to the log files on disk

### On-Disk Structure:

- ❖ System Tablespace
- ❖ Doublewrite Buffer Files
- ❖ Undo Tablespaces
- ❖ Redo Log Files
- ❖ File-Per-Table Tablespaces
- ❖ General Tablespaces
- ❖ Temporary Tablespaces

# MySQL User Administration

- ❖ DBA Account
- ❖ MySQL Permissions
- ❖ **WITH GRANT OPTION**
- ❖ MySQL Workbench
- ❖ MySQL Roles
- ❖ Difference between **Roles** & **Users**
- ❖ Granting Permissions -> Roles, Roles -> Users
- ❖ Expired Account, Unlock Account
- ❖ Explore **mysql.user** table

# MySQL Configuration

## Option Files:

- ❖ Also called MySQL **Configuration Files**
- ❖ Most MySQL programs can read **startup options** from **option files** (configuration)
- ❖ Convenient way to specify commonly used options so need not to specify on command-line
- ❖ mysqld, mysqladmin, mysqlimport, mysqldump, mysql - examples of MySQL programs
- ❖ **program -verbose -help** - To get which default option file this programs uses
- ❖ Any program starts with **-no-defaults** option reads no option file other than **.mylogin.cnf**



# MySQL Backup & Recovery

What do you want to protect?

- ❖ MySQL Instance - **Physical Backup**
- ❖ Option Files/Configuration Files - **Source Control**
- ❖ Database(s) - **Logical Backup**
- ❖ Table(s) - **Logical Backup**

# MySQL Upgrades

## Upgrade Types:

- ❖ Minor MySQL Version Upgrade
- ❖ Major MySQL Version Upgrade

# Popular MySQL Servers

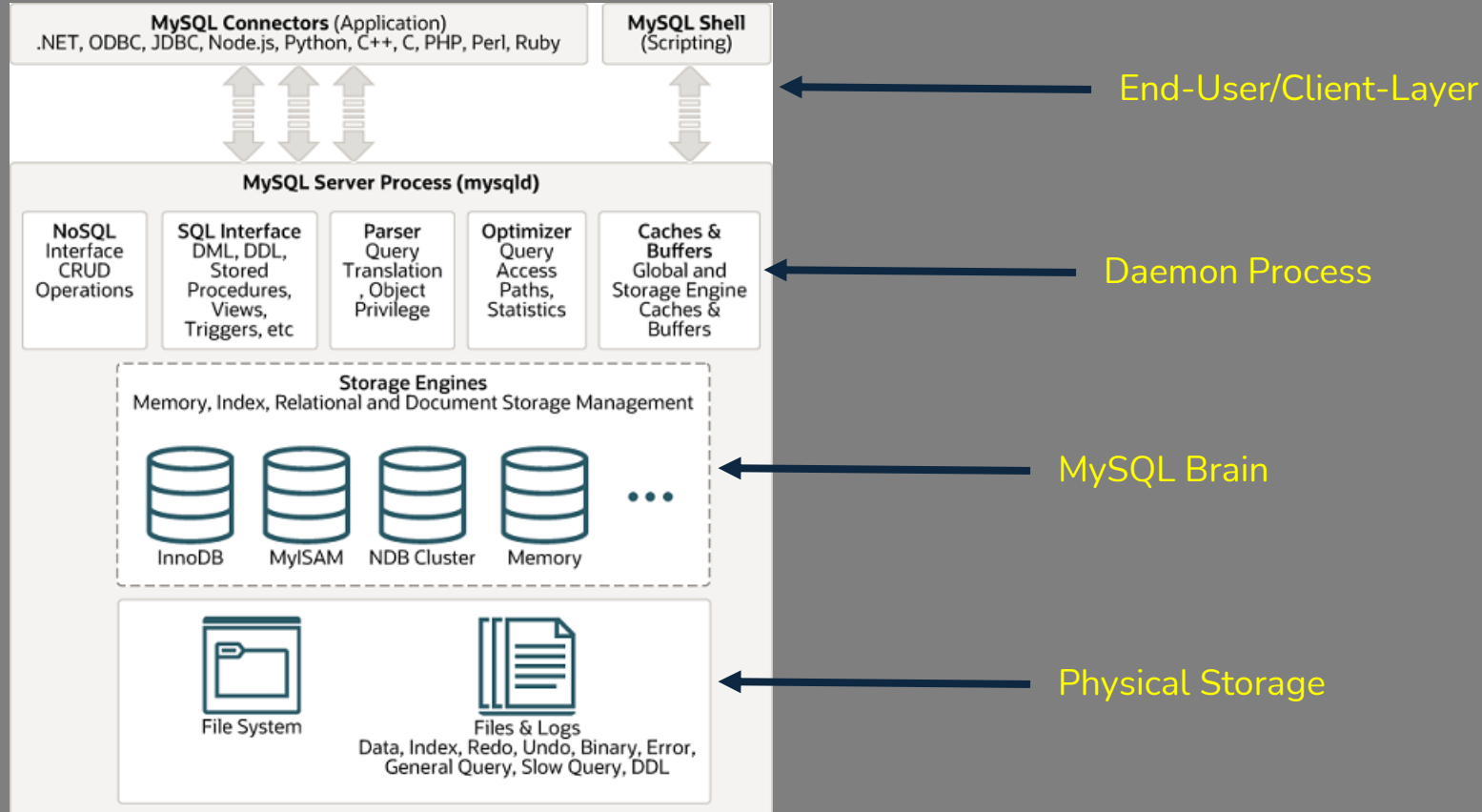
Oracle MySQL, MariaDB, Percona MySQL are **forks** of original MySQL

**MariaDB Provides:** Columnar Storage, Temporal Tables, Transaction replay, non-blocking backups, Oracle Compatible, Pluggable Storage Engines

**Percona Provides:** XtraDB Cluster, Percona-Toolkit, XtraBackup, TokuDB, MyRocks Storage Engine, InnoDB Full-Text Search, Compressed Columns

**Oracle MySQL Provides:** HeatWave, mysqlcheck, Scalability, Dual Passwords, High Performance

# MySQL Architecture



# MYSQL File Types

Installed MySQL Files Location :

## DATA DIRECTORY:

- ❖ Also known as **datadir**
- ❖ Default location: **/var/lib/mysql**
- ❖ Owned by **mysql** os user as it's home directory
- ❖ All the new **databases** that we create, reside there as **folder**

## LOG FILES:

- ❖ Default location is **/var/log/mysql/error.log**
- ❖ Very critical file and single source of truth for all errors, warnings, info etc
- ❖ Also contains initial **root** password - newly installed MySQL

## GLOBAL CONFIGURATION FILE:

- ❖ Default location is **/etc/mysql/my.cnf**
- ❖ Contains all the configuration settings that will be loaded when server starts

# MYSQL Executable Programs

mysql	mysqladmin	mysqlbinlog	mysqlcheck	mysql_config_editor
mysqld_pre_systemd	mysqld_safe	mysql_ssl_rsa_setup	mysql_tzinfo_to_sql	mysql_upgrade
mysqlslap	mysql_secure_installation	mysqlshow	mysqld_safe	mysqldump

# MYSQL Shell Commands

## help:

- ❖ \h or \?
- ❖ Prints help about MySQL Shell and all available shell commands
- ❖ Display help for any of the shell commands

## quit:

- ❖ Quits or Exits from MySQL Shell - \q

## status:

- ❖ Shortcut is \s
- ❖ For how long MySQL Server has been up, what is my connection id, version of MySQL
- ❖ Is the current user logged in locally or from a remote location

## system:

- ❖ Shortcut is \!
- ❖ Run operating system commands within MySQL Shell

# MYSQL Shell Commands

## use:

- ❖ \u for short
- ❖ Use another database
- ❖ Takes database name as argument

## source:

- ❖ \. - Execute SQL file (.sql extension)
- ❖ Takes SQL file name as argument

## edit:

- ❖ Edit the SQL statement that you recently executed



# MYSQL Socket File

mysql.sock:

- ❖ MySQL special file that manages **connections** to the mysql server
- ❖ Used for local clients - if user is on the database host and want to connect to mysql
- ❖ Local clients/users can't connect to MySQL without this file
- ❖ Owned by **mysql** user and default location is **/var/run/mysqld**
- ❖ This special connection = UNIX socket - Remote connection = TCP/IP
- ❖ If file is empty but mysql creates another file **mysqld.sock.lock** and add pid

# MYSQL Global Variables

## Global Variables:

- ❖ MySQL server maintains many **system** variables that are used to configure how MySQL should operate.
- ❖ **GLOBAL scope & SESSION scope**
- ❖ Global variables affect the overall operation of MySQL server
- ❖ Each Global variable has **default** value which is initialized when server starts
- ❖ Default value can be changed in **option file** or on **command line**
- ❖ Identified by @@ sign
- ❖ SHOW GLOBAL <variable\_name>; or SELECT @@<variable\_name>;
- ❖ Examples of system variables:
  - max\_connections
  - server\_id
  - sql\_mode

# MYSQL Session Variables

## Session Variables:

- ❖ MySQL server maintains many **system** variables that are used to configure how MySQL should operate.
- ❖ **GLOBAL scope & SESSION scope**
- ❖ SESSION variables affect only the **current session**
- ❖ Default value for session variables can only be changed on **command line**
- ❖ Identified by @@ sign
- ❖ SHOW SESSION VARIABLES LIKE <variable\_name>; or SELECT @@<variable\_name>;
- ❖ Examples of session system variables:
  - sql\_mode

# MYSQL SHOW Command

## SHOW Statements:

- ❖ SHOW DATABASES;
- ❖ SHOW TABLES LIKE '%view%';
- ❖ SHOW BINARY LOGS;
- ❖ SHOW BINLOG EVENTS;
- ❖ SHOW ENGINES;
- ❖ SHOW CREATE TABLE | USER | DATABASE;
- ❖ SHOW ERRORS;
- ❖ SHOW WARNINGS;
- ❖ SHOW EVENTS;
- ❖ SHOW TRIGGERS;
- ❖ SHOW PROCESSLIST;

Note: SHOW Statements also accepts **LIKE** clause

# MYSQL System Databases

## System Databases:

- ❖ MySQL server comes with some default system databases
  - information\_schema
  - mysql
  - performance\_schema
  - sys
  - Test - Generally deleted by running `mysql_secure_installation`

# MYSQL System Databases

## information\_shema:

- ❖ Each MySQL instance will have information\_schema database
- ❖ Also called System Catalog or Data Dictionary
- ❖ Provides access to **metadata**, that is data about data
- ❖ The tables in this database are **read-only** - they are actually views
- ❖ So no INSERT, UPDATE, DELETE operations

## mysql:

- ❖ Contains tables that store information required by MySQL server
- ❖ Grant information to user accounts, registry of event scheduler, plugins
- ❖ Replication System Tables
- ❖ System tables with timezone information

# MySQL System Databases

## performance\_schema:

- ❖ Inspect internal execution of the server.
- ❖ Primarily focuses on performance data
- ❖ Information about events waits, database locks, memory allocation

## sys:

- ❖ Collection of views, functions, and stored procedures that help MySQL admins to get insight into MySQL database usage.
- ❖ Similar to `performance_schema` but is more user friendly
- ❖ How many total connections a user has established, memory consumption
- ❖ Database host summary about memory, storage, io

# MYSQL Connections

## localhost-connection:

- ❖ localhost
- ❖ root@localhost

## specific-host-connection:

- ❖ Host or IP Address webserver01 or 192.168.10.10
- ❖ app\_user@webserver01

## any-host-connection:

- ❖ %
- ❖ dba@%



# MYSQL Config Editor

## mysql\_config\_editor:

- ❖ Configure Authentication information for connecting to MySQL server
- ❖ Stores authentication credentials in an obfuscated login path file called **.mylogin.cnf - Encrypted**
- ❖ Location: user's home directory - Syntax: **mysql\_config\_editor set --login-path=client --host= --user= --password**
- ❖ login-path is option group that specify which MySQL server to connect and which account to auth
- ❖ By default mysql client reads **[client]** and **[mysql]** groups

# MySQL Config Editor

.mylogin.cnf:

[client]

user = root

password =

host = localhost

[prod]

user = user

password = password

host = proddb01

# MYSQL Admin Program

## mysqladmin:

- ❖ MySQL Server Administration program
- ❖ Client for performing administrative operations:
  - shutdown
  - create <database\_name>
  - current status
  - ping if MySQL is alive
  - Start Replica
  - Stop Replica
- ❖ Syntax: `mysqladmin options command`

## Example:

- ❖ `mysqladmin status`
- ❖ `mysqladmin ping`
- ❖ `mysqladmin create database`
- ❖ `mysqladmin drop database`

# MYSQL Execute SQL Files

## source:

- ❖ From within mysql shell - using `\.` or `source`
- ❖ `mysql> source file.sql` or `mysql> \. file.sql`

## mysql:

- ❖ By running mysql client program and accepting `.sql` file as input
- ❖ `mysql -host=host_name -user=user_name -password= database_name < file.sql`

## shell script:

- ❖ By creating an `executable` shell script and executing it
- ❖ `mysql -host=host_name database_name < $1`

## pipe method:

- ❖ `cat filename.sql | mysql`

# MYSQL Execute SQL Files

Execute employees.sql - Create staff table in employees db

## Syntax:

- ❖ `mysql>source employees.sql`
- ❖ `mysql -host=localhost employees < employees.sql`
- ❖ `bash employees.sh employees.sql`
- ❖ `cat employees.sql | mysql`

# MYSQL mysqlimport

## mysqlimport:

- ❖ mysqlimport is a data import program
- ❖ Takes .txt with tab-delimited file as input

## Syntax:

- ❖ `mysqlimport [options] database file1.txt [file2.txt] ...`

## Import Data Directory Configuration:

- ❖ `secure_file_priv` - denoted a directory from which data files can be loaded

# MySQL mysqlimport

Load data from staff.txt - Populate staff table in employees db

## Steps:

- ❖ We will use **mysqlimport** utility
- ❖ Fetch **secure\_file\_priv** value
- ❖ Copy file and change permissions
- ❖ Load Data mysqlimport [options] db\_name \${secure\_file\_priv}/staff.txt

# MYSQL mysqlcheck

## mysqlcheck:

- ❖ mysqlcheck is a table maintenance program
- ❖ It checks, repairs, optimize, or analyze tables
- ❖ Table name as input

## Note:

- ❖ Table will be **locked** while mysqlcheck is running - no db operations

## Syntax:

- ❖ `mysqlcheck [options] db_name table_name`



# MySQL mysqlcheck

Check the integrity of staff table

## Steps:

- ❖ We will use **mysqlcheck** utility
- ❖ `mysqlcheck employees staff`

# MYSQL mysqlshow

## mysqlshow:

- ❖ Display database, table, and column information
- ❖ Takes database name and table name as input

## Syntax:

- ❖ `mysqlshow [options] db_name table_name`
- ❖ `mysqlshow [options] db_name table_name [column_name]`

# MYSQL Timezone Data

## mysql\_tzinfo\_to\_sql:

- ❖ Loads the time zone data from zoneinfo database into **system mysql** database
- ❖ Zoneinfo database is actually zone files that describe time zones
- ❖ Typical location on Linux is **/usr/share/zoneinfo**

## Timezone Tables:

- ❖ Time\_zone
- ❖ Time\_zone\_name
- ❖ Time\_zone\_transition
- ❖ Time\_zone\_transition\_type
- ❖ time\_zone\_leap\_second

## Syntax:

- ❖ `mysql_tzinfo_to_sql zoneinfo_database | mysql [options] db_name`

# MySQL Timezone Data

Load Timezone Data into MySQL

## Steps:

- ❖ We will use `mysql_tzinfo_to_sql` utility
- ❖ `mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql mysql`

# MYSQL Example Databases

## MySQL Example Databases:

- ❖ Free to download and use

## Example Databases:

- ❖ employees
- ❖ world
- ❖ sakila

## URL Information:

- ❖ <https://dev.mysql.com/doc/index-other.html>

# MYSQL Storage Engines

FEDERATED	InnoDB	MyISAM	ARCHIVE
BLACKHOLE	CSV	MEMORY	PERFORMANCE_SCHEMA

- Pluggable storage engine architecture - load and unload on a running MySQL Server
- show engines - which storage engine your server support
- Shared library location for all the plugins - plugin\_dir variable
- Can install & uninstall more storage engines

# MYSQL Storage Engines

## FEDERATED STORAGE ENGINE:

- ❖ **Disabled** by default
- ❖ Table created with FEDERATED Storage Engine, normally points to a table in another MySQL instance installed on a separate server.
- ❖ **Linked Server** - Microsoft SQL Server
- ❖ **Database Link** - Oracle
- ❖ Both tables should have the same name and definition
- ❖ The table in requester acts like a **view**
- ❖ Target table can have different storage engine but requester table should be created with FEDERATED

## Syntax:

```
Create table employee_salaries (  
Employee_id int,  
Employee_salary int  
) ENGINE = FEDERATED  
CONNECTION = 'mysql://db_user@target-server:3306/employees/employee_salaries';
```

# MYSQL Storage Engines

## MEMORY STORAGE ENGINE:

- ❖ Called **HEAP** in older versions
- ❖ Very useful for **temporary** tables
- ❖ MEMORY will write table data in **memory**
- ❖ Not Persistent - Data lost on server crash
- ❖ Very fast data retrieval but memory is volatile so use only for read-only cache data or temp tables

## USE CASE:

- ❖ Static Tables - lookup
- ❖ Temporary Tables

## Caveats:

- ❖ No **Transactions** support
- ❖ No Referential Integrity support - No FK
- ❖ NO TEXT data type support - No BLOB column



# MySQL Storage Engines

Create **continents** table in **MEMORY** - world database

## Steps:

- ❖ Create table specifying **ENGINE=MEMORY**
- ❖ Insert data
- ❖ Verify data and also table definition from `information_schema.tables`
- ❖ Restart MySQL Service and observe table is there but data is gone

# MYSQL Storage Engines

## BLACKHOLE STORAGE ENGINE:

- ❖ Acts as a black hole, whatever goes into it, never comes back
- ❖ You can store as much data as you want, when you retrieve it, it returns empty result set
- ❖ Anything you write to it, disappears
- ❖ Does not support transactions

## Syntax:

```
Create table employee_salaries (  
Employee_id int,  
Employee_salary int  
) ENGINE = BLACKHOLE;
```

# MySQL Storage Engines

Create **continents** table in **BLACKHOLE** - world database

## Steps:

- ❖ Create table specifying **ENGINE=BLACKHOLE**
- ❖ Insert data
- ❖ Verify empty result set will return

# MYSQL Storage Engines

## CSV STORAGE ENGINE:

- ❖ Stores table in text files using comma-separated values format
- ❖ MySQL creates a **.csv** file in the \$DATA\_DIR - plain text file
- ❖ CSV format can be read, written by spreadsheet applications like **Excel**
- ❖ Does not support transactions
- ❖ CSV files are not indexed

## USE CASE:

- ❖ When data need to be shared with other applications that also use CSV format

## Syntax:

```
Create table continents (  
cid int NOT NULL,  
cname VARCHAR(25) NOT NULL  
) ENGINE = CSV;
```

# MySQL Storage Engines

Create **continents** table in **CSV** - world database

## Steps:

- ❖ Create table specifying **ENGINE=CSV**
- ❖ Insert data
- ❖ Search for continents.CSV file under \$DATA\_DIR/world

# MYSQL Storage Engines

## MyISAM STORAGE ENGINE:

- ❖ MyISAM = My + ISAM = Indexed Sequential Access Method
- ❖ Indexing algorithm developed by IBM that allows retrieving information from large sets of data in a fast way
- ❖ MyISAM was default storage engine up until MySQL 5.5 - around 2009-2010
- ❖ Good speed advantages especially useful in Data warehouse scenario
- ❖ Replaced by InnoDB
- ❖ Does not support transactions - ACID Model

## USE CASE:

- ❖ Data Warehouse - a lot of reads

# MySQL Storage Engines

Create **continents** table in **MyISAM** - world database

## Steps:

- ❖ Create table specifying **ENGINE=MyISAM**
- ❖ Insert data
- ❖ Start Transaction, Commit, Rollback

# MYSQL Storage Engines

## ARCHIVE STORAGE ENGINE:

- ❖ Produces special-purpose tables that store large amounts of **un-indexed** data in very small footprint
- ❖ Creates .ARZ files with same name as table name
- ❖ ARZ files are binary data files and are called MySQL Archive Storage Engine Data File
- ❖ Uses **gzip** to compress rows

## CAVEAT:

- ❖ No DELETE or UPDATE operation
- ❖ No Partitioning

## Syntax:

```
Create table continents (  
cid int NOT NULL,  
cname VARCHAR(25) NOT NULL  
) ENGINE = ARCHIVE;
```



# MySQL Storage Engines

Create **continents** table in **ARCHIVE** - world database

## Steps:

- ❖ Create table specifying **ENGINE=ARCHIVE**
- ❖ Insert data
- ❖ Look for .ARZ file

# MYSQL Storage Engines

## InnoDB STORAGE ENGINE:

- ❖ ACID compliant storage engine that support all types of transactions
- ❖ A - Atomicity, involves transactions COMMIT & ROLLBACK
- ❖ C - Consistency, mechanism for crash recovery
- ❖ I - Isolation, different isolation levels that applies at each transaction level
- ❖ D - Durability, storage engine interacts with underlying hardware to provide best performance
- ❖ Default storage engine, robust, fast, heart of MySQL
- ❖ Best for OLTP - Online Transaction Processing
- ❖ Row-level locking, indexing
- ❖ InnoDB maintains its own **buffer pool** (memory area where InnoDB cache table and indexed data)

# MySQL Storage Engines

Create **continents** table in **InnoDB** - world database

## Steps:

- ❖ Create table specifying **ENGINE=InnoDB** or skip
- ❖ Insert data
- ❖ Test all operations

# REVIEW CONTENT

- ❖ Storage Engines
- ❖ InnoDB - Default Storage Engine
- ❖ Storage Engine Status
- ❖ Migrate table from one storage engine to other
- ❖ Disable storage engine

# MySQL User Administration

- ❖ DBA Account
- ❖ MySQL Permissions
- ❖ **WITH GRANT OPTION**
- ❖ MySQL Workbench
- ❖ MySQL Roles
- ❖ Difference between **Roles** & **Users**
- ❖ Granting Permissions -> Roles, Roles -> Users
- ❖ Expired Account, Unlock Account
- ❖ Explore **mysql.user** table

# MySQL User Administration

## MySQL Permissions:

- ❖ Permissions are Privileges Granted to MySQL Account to perform actions
- ❖ ALL - All Permissions
- ❖ ALTER, DROP, CREATE - Database, table, index, etc
- ❖ DROP - Database, table, index, etc
- ❖ EXECUTE - Stored Procedure
- ❖ INSERT, DELETE, UPDATE, RENAME - On tables
- ❖ \* - On all objects
- ❖ SELECT, SHOW - Read-Only Permissions
- ❖ Replication Client, Replication Slave
- ❖ Grant Permission(s)

# MySQL User Administration

## WITH GRANT OPTION:

- ❖ Clause used when creating new user
- ❖ Ability to grant others permissions
- ❖ DBA user statement should add this clause
- ❖ Create user with all privileges and with grant option

# MySQL Auth Plugins

Authentication Plugins: `mysql_native_password` & `caching_sha2_password`

`mysql_native_password`:

- ❖ Implements Native Pluggable Authentication
- ❖ NPA is based on the password hashing method in use before the intro of pluggable authentication
- ❖ `mysql_native_password` is not pluggable so there is no library file and this plugin is built-in
- ❖ MySQL 5.7 and older version - `mysql_native_password` was default
- ❖ `mysql --default-auth=mysql_native_password`



# MySQL Auth Plugins

Authentication Plugins: `mysql_native_password` & `caching_sha2_password`

`caching_sha2_password`:

- ❖ Default password authentication plugin starting MySQL 8
- ❖ MySQL recommends using `caching_sha2_password` as preferred plugin
- ❖ The server assigns this plugin to the account and uses it to encrypt the password using **SHA-256**, storing those values in the **plugin** and **authentication\_string** column of **mysql.user** system table
- ❖ Built into the server, need not be loaded explicitly and can't be disabled by unloading it

# MySQL Auth Plugins

Authentication Plugins: `mysql_native_password` & `caching_sha2_password`

which one?

Authentication plugin '`caching_sha2_password`' cannot be loaded

```
ALTER USER john IDENTIFIED WITH 'mysql_native_password' by 'password';
```

```
[mysqld]
```

```
default_authentication_plugin=caching_sha2_password
```

```
CREATE USER john WITH 'caching_sha2_password' by 'password';
```

```
CREATE USER john by 'password';
```

# MySQL User Administration

## MySQL Roles:

- ❖ Named collections of privileges
- ❖ GLOBAL CREATE ROLE or CREATE USER privilege
- ❖ Written in **binary log** when succeeded
- ❖ A Role when created is **locked**, assigned default auth plugin
- ❖ No password i.e authentication\_string is **empty**
- ❖ Roles are considered Users in the **mysql.user** system table

## Syntax:

- ❖ CREATE ROLE IF NOT EXISTS 'reader', 'writer', 'admin';

# MySQL User Administration

- ❖ Create Roles **reader**, **writer**, **admin**
- ❖ Create User **db\_reader**, **db\_writer**, **db\_admin**
- ❖ **Permissions:**
  - **reader**=SELECT on **continents** table
  - **writer**=INSERT, UPDATE, DELETE on continents table
  - **admin**=ALL Permissions on **world** database
- ❖ **Grant:** **reader** to **db\_reader**, **writer** to **db\_writer**, **admin** to **db\_admin**

# MySQL Configuration

## Option Files:

- ❖ Also called MySQL **Configuration Files**
- ❖ Most MySQL programs can read **startup options** from **option files** (configuration)
- ❖ Convenient way to specify commonly used options so need not to specify on command-line
- ❖ mysqld, mysqladmin, mysqlimport, mysqldump, mysql - examples of MySQL programs
- ❖ **program -verbose -help** - To get which default option file this programs uses
- ❖ Any program starts with **-no-defaults** option reads no option file other than **.mylogin.cnf**

## Option Files Format:

# MySQL Configuration

## Option Files Syntax:

- ❖ Comments - # sign
- ❖ option group - stanza
- ❖ option = value
- ❖ Space is allowed either side
- ❖ Value can be without quote, single-quote, double-quote
- ❖ Any option that may be given at command-line, can be given in option file as well
- ❖ -server-id at command-line can be given in option file as server-id
- ❖ option IS NOT variable

## Groups:

- ❖ mysqld, mysqladmin, client, mysql, server

**Note:** [client] option group is read by all client-programs except mysqld

# MySQL Configuration

## Option File Inclusions:

- ❖ `!include` = for file
- ❖ `!includedir` = for multiple option files

## Examples:

- ❖ `!include /home/john/my-options.cnf`
- ❖ `!includedir /home/john`

## Note:

- ❖ All option files must end with `.cnf`

# MySQL Configuration

## MySQL Data Directory:

- ❖ Default path is `/var/lib/mysql`
- ❖ `mysql` user is created and `/var/lib/mysql` set as `home directory`
- ❖ Owned by `mysql` user
- ❖ Should be on its own Filesystem - `SSSD` is preferred
- ❖ Controlled by `datadir` option in `my.cnf`



# MySQL Configuration

- ❖ Move Data Directory to `/var/lib/mysql/prod`

## Steps:

- ❖ Shutdown mysql service
- ❖ Create directory and change owner to mysql
- ❖ Move all data files
- ❖ Set option in my.cnf
- ❖ Restart mysql service

# MySQL Configuration

## MySQL Binary Log Files:

- ❖ Record database changes as events
- ❖ Binary format - **Encrypted**
- ❖ **mysqlbinlog** utility to read binary log files and output in clear text
- ❖ Contains information on how long each statement took that updated data
- ❖ Very import for **Replication** - provides a records of data changes on source
- ❖ **Point-in-Time** Recovery - Bring database up to date from the point of backup
- ❖ After a backup has been restored, the events in the binary log that were recorded after the backup was made are re-executed
- ❖ Default size is **1GB** - controlled by **max\_binlog\_size**
- ❖ **Retention** - How many days worth of binary logs should we keep - **binlog\_expire\_logs\_seconds**

# MySQL Configuration

## Enable Binary Logging:

- ❖ Enabled by default - system variable `log_bin` ON
- ❖ `log_bin` - binlog | mysqld-bin | prod-bin
- ❖ `log_bin_index` - binlog.index | mysqld-bin.index | prod-bin.index

## Disable Binary Logging:

- ❖ `disable-log-bin`

# MySQL Configuration

## TASK:

- ❖ Disable Binary Logging
- ❖ Enable Binary Logging - Move to a new location

# MySQL Configuration

## MySQL Error Log File:

- ❖ Contains a record of mysqld startup and shutdown times
- ❖ Also contains diagnostic messages like errors, warnings that occurs during startup or shutdown, and while the server is running
- ❖ Different MySQL components writes log events in the error log i.e system, innodb, etc
- ❖ **log-error** is the system variable
- ❖ Default error log is **/var/log/error.log**

# MySQL Configuration

- ❖ Configure Error Logging to `/var/log/mysql/errorlog` location

# MySQL InnoDB Storage Engine

## InnoDB Storage Engine:

- ❖ Great general-purpose storage engine that balances high reliability and high performance.
- ❖ MySQL 8, default storage engine
- ❖ It's DML operations follows the ACID model
- ❖ Transactions support commit, rollback, crash-recovery to protect data
- ❖ Row-Level Locking
- ❖ InnoDB tables arrange data on disk to optimize queries based on PK
- ❖ Each InnoDB table has primary key index, clustered index, arrange data
- ❖ Data Integrity - support FK constraints

# MySQL InnoDB Storage Engine

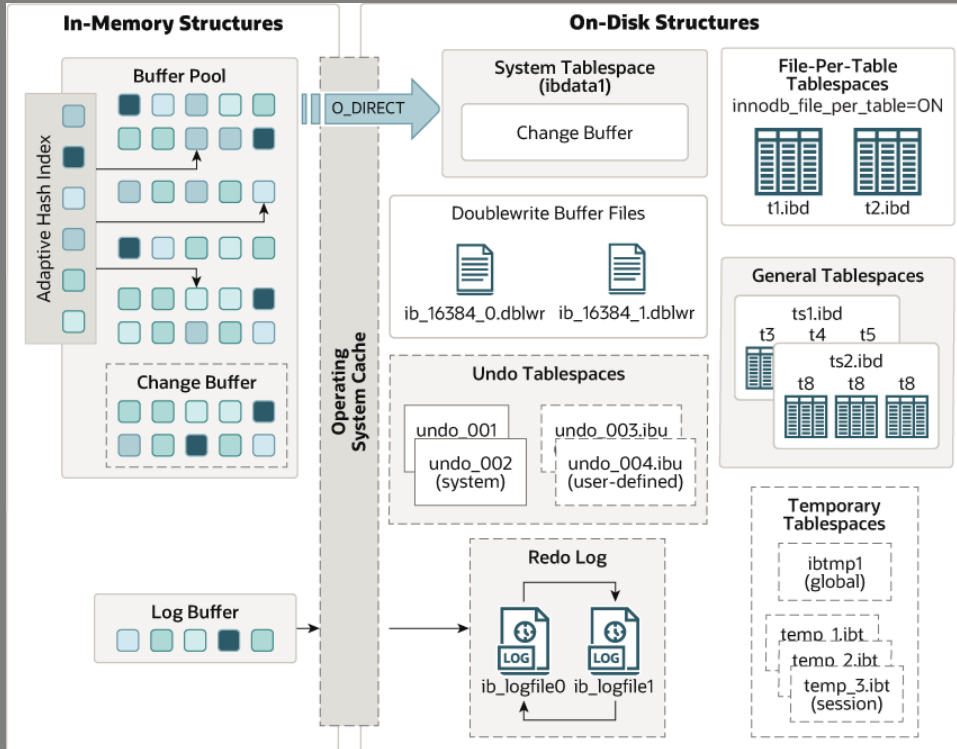
## InnoDB Storage Engine Features:

- ❖ Compress Data
- ❖ Data Caches
- ❖ Encrypted Data
- ❖ Full-Text Search Index
- ❖ Referential Integrity
- ❖ Replication
- ❖ Storage Limit - 64 TB
- ❖ Row Level Locking Granularity



# MySQL InnoDB Architecture

## InnoDB Architecture:



### In-Memory Structure:

- ❖ **Buffer Pool** - Area in main memory where InnoDB caches table & index data as it is accessed
- ❖ **Change Buffer** - caches changes to non-clustered indexes
- ❖ **Adaptive Hash Index** - acts like in-memory db
- ❖ **Log Buffer** - memory area that holds data to be written to the log files on disk

### On-Disk Structure:

- ❖ System Tablespace
- ❖ Doublewrite Buffer Files
- ❖ Undo Tablespaces
- ❖ Redo Log Files
- ❖ File-Per-Table Tablespaces
- ❖ General Tablespaces
- ❖ Temporary Tablespaces

# MySQL InnoDB Storage Engine

## InnoDB Buffer Pool:

- ❖ It is area of main memory that is given to storage engine where it caches the table and index data as it is accessed
- ❖ `innodb_buffer_pool_size`
- ❖ `innodb_buffer_pool_instances` - 1 for small
- ❖ `innodb_buffer_pool_chunk_size` - 128MB default
- ❖ `innodb_buffer_pool_size = innodb_buffer_pool_chunk_size * innodb_buffer_pool_instances`

## InnoDB Buffer Pool Status:

- ❖ `show engine innodb status`

# MySQL InnoDB Architecture

## InnoDB Log Buffer:

- ❖ InnoDB log buffer allows transactions to run without committing before writing to the log files on disk.
- ❖ `Innodb_log_buffer_size` - system variable
- ❖ Bigger log buffer size can accommodate big transactions to save disk I/O
- ❖ Default size is **16MB**

## InnoDB Log Buffer Too Small?

- ❖ `innodb_log_waits` - Number of times that the log buffer was too small
- ❖ A wait is required for it to be flushed before continuing

# MySQL InnoDB Architecture

Check the InnoDB **Log Buffer Size** and see how many times this size was not big enough for transactions

## Steps:

- ❖ innodb\_log\_buffer\_size
- ❖ innodb\_log\_waits

# MySQL InnoDB Architecture

## InnoDB Flush Methods:

- ❖ InnoDB performs certain tasks in background, including flushing of dirty pages from the buffer pool - modified pages that are not yet written to the data files on disk
- ❖ `innodb_flush_method` - system variable
- ❖ `fsync` - Default flush method, flush data, metadata, and log files - causes **double buffering**
- ❖ `O_DSYNC` - flush only data files but causes double-buffering
- ❖ `O_DIRECT` - flush only data files, uses fsync with no double-buffering, read-write directly goes to disk
- ❖ `O_DIRECT_NO_FSYNC` - `O_DIRECT` but skip fsync, not good for **XFS** FS

# MySQL InnoDB Architecture

Change InnoDB Flush Method to **O\_DIRECT**

**Steps:**

- ❖ innodb\_flush\_method

# MySQL InnoDB Architecture

## InnoDB Doublewrite Buffer:

- ❖ Storage area where InnoDB write pages flushed from the buffer pool before writing the pages to their proper positions in the InnoDB data files
- ❖ Implemented to recover from half-written pages
- ❖ In case of OS error, storage issue, unexpected mysqld process exit in the middle of a page write, InnoDB can find a good copy of page from doublewrite buffer during crash recovery
- ❖ Data is written twice, the doublewrite buffer does not require twice as much I/O overhead
- ❖ Prior to MySQL 8.0.20, doublewrite buffer was part of InnoDB system tablespace **ibdata1**
- ❖ As of MySQL **8.0.20**, the doublewrite buffer storage area is located in doublewrite files

# MySQL InnoDB Architecture

## InnoDB Flush Logs Transaction Commit:

- ❖ `innodb_flush_log_at_trx_commit` - system variable - controls balance between strict ACID compliance for commit operations and higher performance - I/O related
- ❖ **1** - default full ACID compliance - Logs are written and flushed to disk at each transaction commit
- ❖ **0** - Logs are written and flushed to disk once per second - transactions for which log have not been flushed can be lost in crash
- ❖ **2** - Logs are written after each transaction commit and flushed to disk once per second - transactions for which log have not been flushed can be lost in crash

## Which value to use?

- ❖ **1** = Safest full ACID compliance - no data loss



# MySQL InnoDB Architecture

## InnoDB Redo Log Files:

- ❖ All about **data recovery** by InnoDB Storage Engine
- ❖ Disk-Based data structure used during **crash recovery** to correct data written by incomplete transactions
- ❖ Any transaction that were active at the time of unexpected exit or fast shutdown
- ❖ The redo log is physically represented on disk by redo log files
- ❖ **ib\_logfile0** & **ib\_logfile1**
- ❖ innodb\_log\_file\_size - default **50MB**
- ❖ innodb\_log\_files\_in\_group - **default 2**
- ❖ innodb\_log\_group\_home\_dir - default to DATA DIR
- ❖ **innodb\_fast\_shutdown** - SET GLOBAL innodb\_fast\_shutdown = 0; **default=1**
  - 0 = Clean Shutdown - Does additional flushing operations, longer time to shutdown but saved time on startup
  - 1 = Fast Shutdown - Shutdown MySQL but read redo log files on startup

# MySQL Backup & Recovery

## mysqldump?

- ❖ Client utility to performs logical backups
- ❖ Produces a set of SQL statements that can be executed to reproduce original database object
- ❖ Can backup whole database my skipping some tables also
- ❖ Can take table backup with **where** clause
- ❖ One or more databases can be dump at same time

## Syntax:

- ❖ `mysqldump [options] db_name [tbl_name] > bkup_name.sql`
- ❖ `mysqldump [options] db_name [tbl_name] -where="condition" > bkup_name.sql`
- ❖ `mysqldump [options] db_name -ignore-table=db.tbl_name > bkup_name.sql`
- ❖ `mysqldump [options] -databases db1 db2... > bkup_name.sql`
- ❖ `mysqldump [options] -all-databases > bkup_name.sql`

# MySQL Backup & Recovery

## MySQL Hot Backup:

- ❖ Also called **Physical Backup**
- ❖ Physically copy database files to a backup device while MySQL is online
- ❖ Suitable for critical always-on production applications
- ❖ Best of InnoDB tables - Transactions

## Hot Backup Tools?

- ❖ **mysqlbackup** - Oracle
- ❖ **mariabackup** - MariaDB
- ❖ **xtrabackup** - Percona

# MySQL Backup & Recovery

## mysqlbackup:

- ❖ Enterprise backup tool - **MySQL Enterprise Backup component** - not-free

## mariabackup:

- ❖ Open-source, free provided by MariaDB
- ❖ **Forked copy** of well-known and commonly used backup tool XtraBackup
- ❖ Supports all the main features of Percona XtraBackup

## Xtrabackup:

- ❖ Open-source hot backup tool for MySQL, from Percona
- ❖ Does not lock database during backup
- ❖ Can backup seamless without disrupting the performance

# MySQL Backup & Recovery

## Xtrabackup:

- ❖ Free of cost, production grade hot backup tool
- ❖ No license is required
- ❖ Completely separate from mysqlbackup or InnoDB hot backup
- ❖ Download from <https://www.percona.com/software/mysql-database/percona-xtrabackup>
- ❖ Compatible with on-prem as well as in the cloud
- ❖ Enterprise ready, can be automated
- ❖ Point-in-time Recovery

# MySQL Replication

## Replication:

- ❖ Enables data copy from source server to destination server
- ❖ Source is called **Primary** or Master
- ❖ Destination is called Replica or Slave
- ❖ Primary and **Replica** should be on separate servers
- ❖ Replication is **Asynchronous** by default
- ❖ Replica does not need to be connected to Primary at any given time
- ❖ Replication Selection - All databases, selected, or even selected tables
- ❖ Replication provides **high-availability**

# MySQL Replication

## Replication Methods:

- ❖ **Traditional** - binary log file position based replication
- ❖ **GTID** - Global Transaction Identifier

## Binary Log File Position Based Replication:

- ❖ Replicating events from primary binary log file
- ❖ Requires the log files and positions to be synced between primary and replica

## GTID Based Replication:

- ❖ Newer method, does not require working with binary logs and positions within files
- ❖ GTID replication guarantee consistency between primary and replica as long as transactions committed on primary also been applied to replica
- ❖ Recommend method for replication

# MySQL Replication

## Replication Format:

- ❖ Replication works because events written to the binlog are read from source and then processed on replica
- ❖ Events are recorded in the Binary Log File in different formats according to the type of event
- ❖ binlog\_format is the system global variable that defines which format to use
- ❖ **Statement Based Replication** - SBR
  - binlog\_format = STATEMENT
- ❖ **Row Based Replication** - RBR
  - binlog\_format = ROW
- ❖ **Mixed Based Replication** - MBR
  - binlog\_format = MIXED
- ❖ Each binary log format has advantages and disadvantages



# MySQL Replication

## Statement Based Replication: `binlog_format = STATEMENT`

- ❖ Replicate entire SQL statements in binary file, this file is copied over to replica
- ❖ Replica execute all these SQL statements
- ❖ Less data written to log files, consumes less storage space for log files
- ❖ Not all statements are replicated as any non-deterministic behavior is difficult to replicate

## Row Based Replication: `binlog_format = ROW`

- ❖ Replicate only the changed rows
- ❖ Source write events in binary log that indicates how individual tables rows are changed
- ❖ All changes are replicated, consume more space

# MySQL Replication

## Generic Replication Setup Requirements:

- ❖ Binary Logging must be enabled on primary
  - log-bin
  - log-bin-index
- ❖ Unique Server ID for both Primary & Replica
  - server-id
- ❖ Dedicated user for replication
  - replicator or any other name
  - Should have proper permissions
    - GRANT REPLICATION SLAVE ON \*.\* TO replicator
- ❖ Binary Log File Format
  - binlog\_format = STATEMENT|ROW|MIXED

# MySQL Replication

## Primary Server Setup:

- ❖ Suggested to create a separate option file i.e **replication.cnf**
- ❖ **Enable Binary Logging**
  - log-bin = /var/log/mysql/binlog/prod-bin
  - log-bin-index = /var/log/mysql/binlog/prod-bin.index
- ❖ **Set Unique Server ID**
  - server-id = 1
- ❖ **Create Dedicated Replication User**
  - Create user replicator IDENTIFIED BY 'password' ;
  - GRANT REPLICATION SLAVE ON \*.\* TO replicator;
- ❖ **Set Binary Log File Format**
  - binlog\_format = MIXED

# MySQL Replication

## Replica Server Setup:

- ❖ Suggested to create a separate option file i.e **replication.cnf**
- ❖ **Enable Relay Logs**
  - relay-log = /var/log/mysql/relay/replica-bin
  - relay-log-index = /var/log/mysql/relay/replica-bin.index
- ❖ **Set Unique Server ID**
  - server-id = 2
- ❖ **Skip Replica to auto-start**
  - skip-replica-start
- ❖ **Replica should be read only**
  - read-only