

MySQL for Database Administrators

Student Guide - Volume I

D61762GC51 | D108205



Author

KimSeong Loh

Technical Contributors

Guilherme Saraiva
Jesper Wisborg Krogh
Jonathon Coombes
Lig Isler-Turmelle
Mirko Ortensi
Ryan Kuan

Editors

Moushmi Mukherjee
Aju Kumar

Graphic Designer

Kavya Bellur

Publishers

Sujatha Nagendra
Pavithran Adka
Syed Ali

5102272020

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1 Introduction to MySQL

Objectives 1-2
Course Goals 1-3
Course Lesson Map 1-5
Introductions 1-6
Classroom Environment 1-7
MySQL Powers the Web 1-8
MySQL Market Share: DB-Engines 2019 1-9
MySQL Enterprise Edition 1-10
Oracle Premier Support for MySQL 1-11
MySQL and Oracle Integration 1-12
MySQL Websites 1-13
Community Resources 1-14
Oracle University: MySQL Training 1-15
MySQL Certification 1-16
Summary 1-17
Practices 1-18

2 Installing and Upgrading MySQL

Objectives 2-2
Topics 2-3
Installation Sequence 2-4
Installing MySQL from Downloaded Packages 2-5
MySQL RPM Installation Files for Linux 2-6
MySQL RPM Installation Process 2-7
MySQL DEB Installation 2-8
Linux Distribution-Specific Repositories 2-9
Installing MySQL by Using a Package Manager 2-10
Adding a Yum Repository 2-11
Configuring Yum Repository Versions 2-12
Adding an APT Repository 2-13
Configuring Repository Versions 2-14
Manually Configuring the APT Repositories 2-15
Installing MySQL on Windows 2-16
Installing on Windows: MySQL Installer 2-17

Installing on Windows: Selecting Products and Features	2-18
Installing on Windows: Product Configuration	2-19
Installing MySQL as a Windows Service	2-20
Installing MySQL from Source	2-21
Installing MySQL from Binary Archive	2-22
Deploying MySQL Server with Docker	2-25
Quiz	2-27
Topics	2-28
Linux MySQL Server Installation Directories	2-29
Windows MySQL Server Installation Directory	2-30
MySQL Programs	2-31
mysqld: MySQL Server Process	2-32
Installation Programs	2-33
Utility Programs	2-34
mysql_config_editor	2-35
.mylogin.cnf Format	2-36
Login Paths	2-37
Command-Line Client Programs	2-38
Launching Command-Line Client Programs	2-39
Topics	2-40
Configuring Mandatory Access Control	2-41
SELinux Example	2-42
AppArmor: Example	2-43
Changing the root Password	2-44
Using mysqladmin to Change the root Password	2-45
Quiz	2-46
Topics	2-47
Starting and Stopping MySQL	2-48
Stopping MySQL with mysqladmin	2-49
MySQL Service Files	2-50
Starting and Stopping MySQL on Windows	2-51
Starting and Stopping MySQL on Windows: MySQL Notifier	2-52
Topics	2-53
Upgrading MySQL	2-54
Reading Release Notes	2-55
MySQL Shell Upgrade Checker Utility	2-56
Using In-Place Upgrade Method	2-57
Using Logical Upgrade Method	2-58
mysql_upgrade	2-59
Summary	2-60
Practices	2-61

3 Understanding MySQL Architecture

Objectives 3-2
Topics 3-3
Architecture 3-4
Client/Server Connectivity 3-5
MySQL Server Process 3-6
Terminology 3-7
Server Process 3-8
Topics 3-9
Connection Layer 3-10
Connection Protocols 3-11
Local and Remote Connection Protocol: TCP/IP 3-12
Local Connection Protocol in Linux: Socket 3-13
MySQL and localhost 3-14
Local Connection Protocols in Windows: Shared Memory and Named Pipes 3-15
SSL by Default 3-16
Connection Threads 3-17
Quiz 3-18
Topics 3-19
SQL Layer 3-20
SQL Layer Components 3-21
SQL Statement Processing 3-22
Topics 3-23
Storage Layer 3-24
Storage Engines Provided with MySQL 3-25
Storage Engines: Function 3-26
SQL and Storage Layer Interactions 3-27
Features Dependent on Storage Engine 3-28
InnoDB Storage Engine 3-30
MyISAM Storage Engine 3-31
MEMORY Storage Engine 3-32
ARCHIVE Storage Engine 3-33
NDBCluster Storage Engine 3-34
BLACKHOLE Storage Engine 3-35
Storage Engines Feature Summary 3-36
How MySQL Uses Disk Space 3-38
Data Directory 3-39
Topics 3-40
What Is a Data Dictionary? 3-41
Types of Metadata 3-42

Data Dictionary in Earlier Versions of MySQL 3-43
Transactional Data Dictionary in MySQL 8 3-44
Transactional Data Dictionary: Features 3-45
Serialization of the Data Dictionary 3-46
Dictionary Object Cache 3-47
Topics 3-48
InnoDB Tablespaces 3-49
InnoDB System Tablespace 3-50
File-per-Table Tablespaces 3-51
General Tablespaces 3-52
Choosing Between File-Per-Table and General Tablespaces 3-53
Locating Tablespaces Outside the Data Directory 3-54
Temporary Tablespaces 3-55
Topics 3-56
Redo Logs 3-57
Undo Logs 3-59
Undo Tablespaces 3-61
Temporary Table Undo Log 3-62
Quiz 3-63
Topics 3-64
How MySQL Uses Memory 3-65
Global Memory 3-67
Session Memory 3-68
Log Files and Buffers 3-69
InnoDB Buffer Pool 3-70
Configuring the Buffer Pool 3-71
Topics 3-72
MySQL Plugin Interface 3-73
MySQL Component Interface 3-74
Summary 3-75
Practices 3-76

4 Configuring MySQL

Objectives 4-2
Topics 4-3
MySQL Configuration Options 4-4
Deciding When to Use Options 4-5
Displaying Configured Server Options 4-6
Option Naming Convention 4-7
Using Command-Line Options 4-8
Topics 4-9

Reasons to Use Option Files	4-10
Option File Locations	4-11
Option Files That Each Program Reads	4-12
Standard Option Files	4-13
Option File Groups	4-14
Option Groups That Each Program Reads	4-15
Option Group Names	4-16
Client Options: Examples	4-17
Writing Option Files	4-18
Option File Contents: Example	4-19
Option Precedence in Option Files	4-20
Loading or Ignoring Option Files from the Command Line	4-21
Loading Option Files with Directives	4-22
Displaying Options from Option Files	4-23
Quiz	4-24
Topics	4-25
Server System Variables	4-26
System Variable Scope: GLOBAL and SESSION	4-27
Dynamic System Variables	4-28
Changing Variable Values	4-29
Persisting Global Variables	4-30
Displaying System Variables	4-31
Viewing Variables with Performance Schema	4-32
Topics	4-34
Launching Multiple Servers on the Same Host	4-35
Settings That Must Be Unique	4-36
mysqld_multi	4-37
mysqld_multi: Example Configuration File	4-38
systemd: Multiple MySQL Servers	4-39
Quiz	4-40
Summary	4-41
Practices	4-42

5 Monitoring MySQL

Objectives	5-2
Topics	5-3
Monitoring MySQL with Log Files	5-4
Log File Characteristics	5-5
Error Log	5-6
Binary Log	5-7
General Query Log	5-8

General Query Log: Example	5-9
Slow Query Log	5-10
Slow Query Log: Logging Administrative and Replicated Statements	5-11
Filtering Slow Query Log Events	5-12
Slow Query Log: Example	5-13
Viewing the Slow Query Log with mysqldumpslow	5-14
mysqldumpslow: Example	5-15
Specifying TABLE or FILE Log Output	5-16
Log File Rotation	5-17
Flushing Logs	5-18
Quiz	5-19
Topics	5-20
Status Variables	5-21
Displaying Status Information	5-22
Monitoring Status with mysqladmin	5-23
Topics	5-24
Performance Schema	5-25
Performance Schema Table Groups	5-26
Configuring Performance Schema	5-27
Performance Schema Setup Tables	5-28
Performance Schema Instruments	5-29
Top-Level Instrument Components	5-30
Accessing Performance Schema Metrics	5-31
The sys Schema	5-32
Using the sys Schema Example 1	5-33
Using the sys Schema: Example 2	5-36
Topics	5-37
MySQL Enterprise Audit	5-38
Installing MySQL Enterprise Audit	5-39
Audit Log File Configuration	5-40
Audit Log Filtering	5-41
Audit Log Filter Definitions	5-42
Audit Log File Format	5-43
Audit Log File: New-Style XML Format	5-44
Audit Log File: JSON Format	5-45
Audit Record Values	5-46
Quiz	5-47
Topics	5-48
MySQL Enterprise Monitor	5-49
Installing MySQL Enterprise Monitor	5-50
Installing the Service Manager	5-51

Post-Installation Configuration 5-53
Installing Agents 5-54
MySQL Enterprise Monitor: Managing Multiple Servers 5-55
MySQL Enterprise Monitor: Timeseries Graphs 5-56
MySQL Enterprise Monitor: Advisors 5-57
MySQL Enterprise Monitor: Events 5-58
Topics 5-59
SHOW PROCESSLIST 5-60
Performance Schema Threads Table 5-61
Killing Processes 5-62
Limiting User Activity 5-63
Setting Resource Limits 5-64
Resetting Limits to Default Values 5-65
Summary 5-66
Practices 5-67

6 Managing MySQL Users

Objectives 6-2
Topics 6-3
Importance of User Management 6-4
Authentication and Authorization 6-5
User Connection and Query Process 6-6
Viewing User Account Settings 6-7
Pluggable Authentication 6-8
Local Connection 6-9
Remote Connection 6-10
Topics 6-11
Account Names 6-12
Host Name Patterns 6-13
Creating a User Account 6-14
Roles 6-15
Creating a Role 6-16
Manipulating User Accounts and Roles 6-17
Topics 6-18
Setting the Account Password 6-19
Dual Password Support 6-20
Expiring Passwords Manually 6-21
Configuring Password Expiration 6-22
Changing Expired Passwords 6-23
Quiz 6-24
Topics 6-25

Pluggable Authentication 6-26
Cleartext Client-Side Authentication Plugin 6-27
Loadable Authentication Plugins 6-28
Enterprise Authentication Plugins 6-29
PAM Authentication Plugin 6-30
Configuring the PAM Authentication Plugin 6-31
Creating Users that Authenticate with PAM 6-32
Creating PAM Proxied Users 6-33
Logging In with PAM Accounts 6-34
Topics 6-35
Authorization 6-36
Determining Appropriate User Privileges 6-37
Privilege Scope 6-38
Granting Administrative Privileges 6-39
Dynamic Privileges 6-40
Special Privileges 6-41
GRANT Statement 6-42
Granting Permissions on Columns 6-43
Granting Roles to Users 6-44
Displaying GRANT Privileges 6-45
Displaying Privileges for Another User 6-46
Displaying Privileges for a Role 6-47
Revoking Account Privileges 6-48
REVOKE: Examples 6-49
Partial Revoke 6-51
User Privilege Restrictions 6-52
Quiz 6-53
Topics 6-54
Using Role Privileges 6-55
Activating Roles at Server Level 6-56
Activating Roles at User Level 6-57
Activating Roles at Session Level 6-58
Mandatory Roles 6-59
Topics 6-60
Grant Tables 6-61
Grant Table Contents 6-62
Use of Grant Tables 6-63
Effecting Privilege Changes 6-64
Summary 6-65
Practices 6-66

7 Securing MySQL

- Objectives 7-2
- Topics 7-3
- Security Risks 7-4
 - MySQL Installation Security Risks 7-5
 - Topics 7-6
 - Securing MySQL from Public Networks 7-7
 - Preventing Network Security Risks 7-8
 - Securing MySQL in Private Networks 7-9
 - Topics 7-10
 - Secure Connections 7-11
 - Secure Connection: Overview 7-12
 - Generating a Digital Certificate 7-13
 - Server Security Defaults 7-14
 - SSL Is Enabled by Default with MySQL Clients 7-15
 - Disabling SSL on MySQL Server 7-16
 - Setting Client Options for Secure Connections 7-17
 - Client --ssl-mode Option: Example 7-18
 - Setting the Permitted Versions for SSL/TLS for the Server 7-19
 - Setting the Permitted Versions for SSL/TLS for the Client 7-20
 - Setting the Cipher to Use for Secure Connections 7-21
 - Global System Variable and Session Status Variables for Ciphers 7-22
 - Cipher System and Status Variables: Example 1 7-23
 - Cipher System and Status Variables: Example 2 7-24
 - Setting Client SSL/TLS Options by User Account 7-25
 - Generating a Digital Certificate 7-26
 - SSL Server Variables for Digital Certificates 7-27
 - SSL Client Options for Digital Certificates 7-28
 - Securing a Remote Connection to MySQL 7-29
 - Quiz 7-30
 - Topics 7-31
 - Preventing MySQL Password Security Risks 7-32
 - How Attackers Derive Passwords 7-33
 - Password Validation Component 7-34
 - Validate Password Component Variables 7-35
 - Changing the Default Password Validation Variables 7-36
 - Other Password Considerations 7-37
 - Locking an Account 7-38
 - Pluggable Authentication 7-39
 - Preventing Application Password Security Risks 7-40

Connection-Control Plugin	7-41
Installing the Connection-Control Plugin	7-42
Monitoring Connection Failures	7-43
Using the CONNECTION_CONTROL_FAILED_LOGIN_ATTEMPTS Plugin	7-44
Quiz	7-45
Topics	7-46
Limiting Operating System Usage	7-47
Limiting Operating System Accounts	7-48
Operating System Security	7-49
File System Security	7-50
Preventing File System Security Risks	7-51
Topics	7-52
Keyring	7-53
Deploying a Keyring	7-54
Key Management Functions	7-55
Encrypting InnoDB Tablespaces	7-56
Encrypting InnoDB Redo Logs and Undo Logs	7-57
InnoDB Encryption Keys	7-58
Encrypting Binary Log and Relay Log	7-59
Binary Log Encryption Keys	7-60
Topics	7-61
Protecting Your Data from SQL Injection Attacks	7-62
SQL Injection: Example	7-63
Detecting Potential SQL Injection Attack Vectors	7-64
Preventing SQL Injection Attacks	7-65
Topics	7-66
MySQL Enterprise Firewall	7-67
Enterprise Firewall Plugins	7-68
Enterprise Firewall Database Components	7-69
Installing MySQL Enterprise Firewall	7-70
Registering Accounts with the Firewall	7-71
Training the Firewall	7-72
Statement Digests	7-73
Enabling Firewall Protection	7-74
Disabling the Firewall	7-75
Monitoring the Firewall	7-76
Quiz	7-77
Summary	7-78
Practices	7-79

8 Maintaining a Stable System

Objectives	8-2
Topics	8-3
Stable Systems	8-4
Measuring What You Manage	8-5
Establishing a Baseline	8-6
Application Profiling	8-7
Topics	8-8
Asking “What Could Go Wrong?”	8-9
Components in a MySQL Server Installation	8-10
Server Hardware	8-11
Problems with Hardware	8-12
Virtualized Environment	8-13
Operating System	8-14
Coexistent Applications	8-15
Network Failures	8-16
Application Failures	8-17
Force Majeure	8-18
Topics	8-19
Capacity Planning	8-20
Monitoring Table Size	8-21
Calculating Logical Size: Data and Indexes	8-22
Calculating Physical Size: Querying Information Schema	8-23
Calculating Physical Size: Reading the File System	8-24
Scalability	8-25
Scaling Up and Scaling Out	8-26
Quiz	8-27
Topics	8-28
Establishing the Nature of a Problem	8-29
Identifying the Problem	8-30
Common Problems	8-31
Resolving Problems	8-32
Topics	8-33
Identifying the Causes of Server Slowdowns	8-34
Investigating Slowdowns	8-35
Quiz	8-36
Topics	8-37
How MySQL Locks Rows	8-38
Identifying Lock Contention	8-39
InnoDB Table Locks	8-40
InnoDB Row Locks	8-41

Troubleshooting Locks by Using SHOW PROCESSLIST 8-42
Monitoring Data Locks with Information Schema and Performance Schema 8-43
Information Schema INNODB_TRX View 8-44
Performance Schema data_locks Table 8-45
Performance Schema data_lock_waits Table 8-47
sys.innodb_lock_waits View 8-48
sys.innodb_lock_waits: Example Query 8-49
Performance Schema metadata_locks Table 8-50
sys.schema_table_lock_waits View 8-51
Topics 8-52
InnoDB Recovery 8-53
Using --innodb_force_recovery 8-54
Summary 8-55
Practices 8-56

9 Optimizing Query Performance

Objectives 9-2
Topics 9-3
Identifying Slow Queries 9-4
Choosing What to Optimize 9-5
Topics 9-6
Using EXPLAIN to See Optimizer's Choice of Index 9-7
EXPLAIN: Example 9-8
EXPLAIN Output 9-9
Common type Values 9-11
Displaying Query Rewriting and Optimization Actions 9-12
EXPLAIN Example: Table Scan 9-13
EXPLAIN Example: Primary Key 9-14
EXPLAIN Example: Non-unique Index 9-15
EXPLAIN and Complex Queries 9-16
EXPLAIN Example: Simple Join 9-17
Explanation of Simple Join Output 9-18
EXPLAIN FORMAT 9-19
EXPLAIN FORMAT: JSON Example 9-20
EXPLAIN ANALYZE 9-21
Hash Join Optimization 9-22
Topics 9-23
Index Types 9-24
Creating Indexes to Improve Query Performance 9-25
Creating Indexes on Existing Tables 9-26
Dropping Indexes on Existing Tables 9-27

Displaying Indexes Metadata 9-28
Invisible Indexes 9-29
Topics 9-30
Maintaining InnoDB Index Statistics 9-31
Automatically Updating Index Statistics 9-32
Using ANALYZE TABLE 9-33
Rebuilding Indexes 9-34
mysqlcheck Client Program 9-35
Histograms 9-36
Example: The Query 9-37
Example: Creating a Histogram 9-38
Example: The Query with Histogram Data 9-39
Topics 9-40
MySQL Query Analyzer 9-41
Query Response Time Index 9-42
Query Analyzer User Interface 9-43
Quiz 9-44
Summary 9-45
Practices 9-46

10 Choosing a Backup Strategy

Objectives 10-2
Topics 10-3
Reasons to Back Up 10-4
Backup Types 10-5
Hot Backups 10-6
Cold Backups 10-7
Warm Backups 10-8
Quiz 10-9
Topics 10-10
Backup Techniques 10-11
Logical Backups 10-12
Logical Backup Conditions 10-14
Logical Backup Performance 10-15
Physical Backups 10-16
Physical Backup Files 10-17
Physical Backup Conditions 10-18
Online Disk Copies 10-19
Snapshot-Based Backups 10-20
Performing a Snapshot 10-21
Replication-Based Backups 10-22

Binary Log Backups 10-23
Binary Logging and Incremental Backups 10-24
Quiz 10-25
Topics 10-26
Comparing Backup Methods 10-27
Deciding a Backup Strategy 10-28
Backup Strategy: Decision Chart 10-29
More Complex Strategies 10-30
Summary 10-31
Practices 10-32

11 Performing Backups

Objectives 11-2
Topics 11-3
Backup Tools: Overview 11-4
Topics 11-5
MySQL Enterprise Backup 11-6
MySQL Enterprise Backup: Storage Engines 11-7
MySQL Enterprise Backup: InnoDB Files 11-8
MySQL Enterprise Backup: Non-InnoDB Files 11-9
Full Backups 11-10
Single-File Backups 11-11
Backup Process 11-12
Incremental Backups 11-13
Differential Backups 11-14
Validate Operations 11-15
Restore Operations 11-16
Restore Commands 11-17
Restoring Incremental Backups 11-18
Update Operations 11-19
Single-File Operations 11-20
Basic Privileges Required for MySQL Enterprise Backup 11-21
Granting Required Privileges 11-22
Quiz 11-23
Topics 11-24
mysqldump and mysqlpump 11-25
mysqldump 11-26
Ensuring Data Consistency with mysqldump 11-27
mysqldump Options for Creating Objects 11-28
mysqldump Options for Dropping Objects 11-29
mysqldump General Options 11-30

Restoring mysqldump Backups 11-31
Using mysqlimport 11-32
Privileges Required for mysqldump 11-33
Privileges Required for Reloading Dump Files 11-34
mysqlpump 11-35
Specifying Objects to Back Up with mysqlpump 11-36
Parallel Processing with mysqlpump 11-37
Quiz 11-38
Topics 11-39
Physical InnoDB Backups: Overview 11-40
Portability of Physical Backups 11-41
Physical InnoDB Backup Procedure 11-42
Recovering from Physical InnoDB Backups 11-43
Using Transportable Tablespaces for Backup 11-44
Transportable Tablespaces: Copying a Table to Another Instance 11-45
Physical MyISAM and ARCHIVE Backups 11-46
Physical MyISAM and ARCHIVE Backup Procedure 11-47
Recovering from Physical MyISAM or Archive Backups 11-48
LVM Snapshots 11-49
LVM Backup Procedure 11-50
Backing Up Log and Status Files 11-51
Topics 11-52
Replication as an Aid to Backup 11-53
Backing Up from a Replication Slave 11-54
Backing Up from Multiple Sources to a Single Server 11-55
Topics 11-56
Processing Binary Log Contents 11-57
Selective Binary Log Processing 11-58
Point-in-Time Recovery 11-59
Configuring MySQL for Restore Operations 11-60
Quiz 11-61
Summary 11-62
Practices 11-63

12 Configuring a Replication Topology

Objectives 12-2
Topics 12-3
MySQL Replication 12-4
Replication Masters and Slaves 12-5
Relay Slaves 12-6
Complex Topologies 12-7

Quiz 12-8
Topics 12-9
Replication Conflicts 12-10
Replication Conflicts: Example Scenario with No Conflict 12-11
Replication Conflicts: Example Scenario with Conflict 12-12
Topics 12-13
Replication Use Cases 12-14
Replication for Horizontal Scale-Out 12-15
Replication for Business Intelligence and Analytics 12-16
Replication for Geographic Data Distribution 12-17
Replicating with the BLACKHOLE Storage Engine 12-18
Replication for High Availability 12-19
Topics 12-20
Configuring Replication 12-21
Configuring Replication Masters 12-22
Configuring Replication Slaves 12-23
CHANGE MASTER TO 12-24
Finding Log Coordinates 12-25
Global Transaction Identifiers (GTIDs) 12-26
Identifying the Source Server 12-27
Logging Transactions 12-28
Replication with GTIDs 12-29
Replication Filtering Rules 12-30
Applying Filtering Rules 12-31
Quiz 12-32
Topics 12-33
Binary Log Formats 12-34
Row-Based Binary Logging 12-35
Statement-Based Binary Logging 12-36
Mixed Format Binary Logging 12-37
Replication Logs 12-38
Crash-Safe Replication 12-39
Topics 12-40
Asynchronous Replication 12-41
Semisynchronous Replication 12-42
Advantages and Disadvantages of Semisynchronous Replication 12-43
Enabling Semisynchronous Replication 12-44
Multi-Source Replication 12-45
Configuring Multi-Source Replication for a GTID-Based Master 12-46
Configuring Multi-Source Replication for a Binary Log Based Master 12-47
Controlling Slaves in a Multi-Source Replication Topology 12-48

Topics	12-49
MySQL Clone Plugin	12-50
Installing the Clone Plugin	12-51
Granting Permissions to Users	12-52
Cloning Local Data	12-53
Cloning Remote Data	12-54
Cloning for Replication	12-55
Clone Plugin Limitations	12-56
Summary	12-57
Practices	12-58

13 Administering a Replication Topology

Objectives	13-2
Topics	13-3
Failover with Log Coordinates	13-4
Potential Problems When Executing a Failover with Log Coordinates	13-5
Avoiding Problems When Executing a Failover with Log Coordinates	13-6
Failover with GTIDs	13-7
Topics	13-8
Replication Threads	13-9
The Master's Binlog Dump Thread	13-10
Single-Threaded Slaves	13-11
Multithreaded Slaves	13-12
Controlling Slave Threads	13-13
Resetting the Slave	13-14
Quiz	13-15
Topics	13-16
Monitoring Replication	13-17
Slave Thread Status	13-18
Master Log Coordinates	13-19
Relay Log Coordinates	13-20
Replication Slave I/O Thread States	13-21
Replication Slave SQL Thread States	13-24
Monitoring Replication by Using Performance Schema	13-26
Replication Tables in Performance Schema	13-27
MySQL Enterprise Monitor Replication Dashboard	13-28
Topics	13-29
Troubleshooting MySQL Replication	13-30
Examining the Error Log	13-32
SHOW SLAVE STATUS Error Details	13-34
Checking I/O Thread States	13-35

Monitoring Multi-Source Replication 13-36
Summary 13-37
Practices 13-38

14 Achieving High Availability with MySQL InnoDB Cluster

Objectives 14-2
Topics 14-3
What Is MySQL InnoDB Cluster? 14-4
Architecture 14-5
MySQL Group Replication Plugin 14-6
How Group Replication Works 14-7
Single-Primary Mode 14-8
Multi-Primary Mode 14-9
Conflict Resolution 14-10
Consensus and Quorum 14-11
Use Cases 14-12
Group Replication: Requirements and Limitations 14-13
Quiz 14-14
Topics 14-15
MySQL Shell (mysqlsh) 14-16
Using MySQL Shell to Execute a Script 14-17
MySQL Router (mysqlrouter) 14-18
Topics 14-19
Deployment Scenarios 14-20
Deploying Sandbox Instances and Creating the Cluster 14-21
Production Deployment 14-22
Distributed Recovery 14-23
Connecting Clients to the Cluster 14-24
Quiz 14-25
Topics 14-26
Managing Sandbox Instances 14-27
Checking the Status of a Cluster 14-28
Viewing the Structure of a Cluster 14-29
Checking the State of an Instance 14-30
Updating a Cluster Metadata 14-31
Removing Instances from the Cluster 14-32
Rejoining an Instance to the Cluster 14-33
Restoring Quorum Loss 14-34
Recovering the Cluster from a Major Outage 14-35
Dissolving a Cluster 14-36
Disabling super_read_only 14-37

Customizing a MySQL InnoDB Cluster	14-38
Customizing an Instance	14-39
Configuring Secure Connection in a Cluster	14-40
Creating a Server Whitelist	14-41
Summary	14-42
Practices	14-43

15 Conclusion

Course Goals	15-2
Oracle University: MySQL Training	15-4
MySQL Websites	15-5
Your Evaluation	15-6
Thank You	15-7
Q&A Session	15-8

1

Introduction to MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Objectives



After completing this lesson, you should be able to:

- Describe the course goals
- List MySQL products and professional services
- Access MySQL information and services from Oracle websites and community resources
- Find information about MySQL courses and certification options



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Course Goals

After completing this course, you should be able to:

- Describe MySQL products and services
- Access MySQL resources
- Install the MySQL server and client programs
- Upgrade MySQL on a running server
- Describe MySQL architecture
- Explain how MySQL processes, stores, and transmits data
- Configure MySQL server and client programs
- Use server logs and other tools to monitor database activity
- Create and manage users and roles
- Protect your data from common security risks



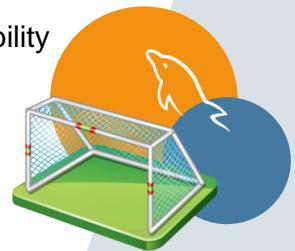
ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Course Goals

After completing this course, you should be able to:

- Maintain a stable system
- Troubleshoot server slowdowns and other common problems
- Identify and optimize poorly performing queries
- Define and implement a backup strategy
- Perform physical and logical backups of your data
- Describe MySQL replication and its role in high availability and scalability
- Configure simple and complex replication topologies
- Administer a replication topology
- Configure and administer InnoDB Cluster



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Course Lesson Map



- Day 1
- 1. Introduction to MySQL
 - 2. Installing and Upgrading MySQL
 - 3. Understanding MySQL Architecture



- Day 2
- 4. Configuring MySQL
 - 5. Monitoring MySQL
 - 6. Managing MySQL Users



- Day 3
- 7. Securing MySQL
 - 8. Maintaining a Stable System



- Day 4
- 9. Optimizing Query Performance
 - 10. Choosing a Backup Strategy
 - 11. Performing Backups



- Day 5
- 12. Configuring a Replication Topology
 - 13. Administering a Replication Topology
 - 14. Achieving High Availability with MySQL InnoDB Cluster
 - 15. Conclusion



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Introductions

- Name
- Company affiliation
- Title, function, and job responsibilities
- Experience related to topics covered in this course
- Reason for enrolling in this course
- Expectations from this course



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Classroom Environment

- Logistics
 - Restrooms
 - Break rooms and designated smoking areas
 - Cafeterias and restaurants in the area
- Emergency evacuation procedures
- Instructor contact information
- Mobile phone usage
- Online course attendance confirmation form



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL Powers the Web

Digital Disruptors and Enterprises Rely on MySQL to Innovate



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The slide lists several digital disruptors and companies that have redefined customer expectations. They rely on MySQL to drive digital transformation initiatives.

They choose MySQL because it is a modern database designed for web-based applications, which allows them to innovate quickly. But those are not the only reasons. As the world's most popular open-source database with over 20 years of existence, MySQL is a proven, battle-tested, and mature technology.

MySQL Market Share: DB-Engines 2019

DB-ENGINES

MySQL database is the second most popular database according to DB-Engines Ranking in October 2019.

MySQL has consistently been in the top three in the last five years.

Rank	Oct 2019	Sep 2019	Oct 2018	DBMS	Database Model	Oct 2019
1.	1.	1.	1.	Oracle 	Relational, Multi-model 	1355.88
2.	2.	2.	2.	MySQL 	Relational, Multi-model 	1283.06
3.	3.	3.	3.	Microsoft SQL Server 	Relational, Multi-model 	1094.72
4.	4.	4.	4.	PostgreSQL 	Relational, Multi-model 	483.91
5.	5.	5.	5.	MongoDB 	Document	412.09

Source: <https://db-engines.com/en/ranking>

ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL is ubiquitous. This is the DB-Engines Database Ranking. DB-Engines ranks databases (300+) according to their popularity, using a variety of sources.

MySQL is the second most popular database in the world, more popular than Microsoft SQL Server. MySQL is also the most popular open-source database. In the last five years, MySQL has consistently been in the top three.

The reason MySQL is so popular is that it performs, it scales, it is easy to use, and as our customers like to say, "it just works."

Method of calculating the scores of the DB-Engines Ranking

The DB-Engines Ranking is a list of database management systems ranked by their current popularity. We measure the popularity of a system by using the following parameters:

- Number of mentions of the system on websites
- General interest in the system
- Frequency of technical discussions about the system
- Number of job offers, in which the system is mentioned
- Number of profiles in professional networks, in which the system is mentioned
- Relevance in social networks

MySQL Enterprise Edition



Advanced Features	Management Tools	Support
<ul style="list-style-type: none">• High Availability• Authentication• Audit• Encryption and TDE• Firewall• Masking	<ul style="list-style-type: none">• Monitoring• Backup• Development• Administration• Migration	<ul style="list-style-type: none">• Technical Support• Consultative Support• Oracle Certifications

ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL Enterprise Edition consists of the following components:

- **MySQL Enterprise Server:** fully integrated transaction-safe, ACID-compliant database with full commit, rollback, crash recovery, and row-level locking capabilities
- **MySQL Enterprise High Availability:** MySQL InnoDB Cluster delivers an integrated, native, HA solution for your databases.
- **MySQL Enterprise Scalability:** MySQL Enterprise Thread Pool allows you to scale the performance of your application in the face of increasing user, query, and data loads.
- **MySQL Enterprise Authentication:** Authenticate MySQL users against your existing directory services and security rules
- **MySQL Enterprise Audit:** Generate a complete audit trail to track MySQL access and usage
- **MySQL Enterprise Encryption:** Protect sensitive data stored in MySQL databases, in backups, or during transfer
- **MySQL Enterprise Transparent Data Encryption (TDE):** Protect data at rest and securely manage your encryption keys
- **MySQL Enterprise Firewall:** Ensure real-time protection against database-specific attacks
- **MySQL Enterprise Masking:** De-identify, anonymize sensitive data
- **MySQL Enterprise Monitor:** Provides real-time visibility into the performance and availability of all your MySQL databases
- **MySQL Enterprise Backup:** Delivers hot, online, secure, non-blocking database backups
- **MySQL Workbench Enterprise Edition:** Provides a GUI to design, develop, and administer MySQL database, as well as to migrate database objects from other relational databases to MySQL

Oracle Premier Support for MySQL

- Largest MySQL engineering and support organization
- Backed by the MySQL developers
- World-class support, in 29 languages
- Hot fixes and maintenance releases
- 24 x 7 x 365
- Unlimited incidents
- Consultative support
- Global scale and reach



Get immediate help for any MySQL issue, plus expert advice.

"The MySQL support service has been essential in helping us with troubleshooting and providing recommendations for the production cluster. Thanks."

– Carlos Morales (Playfulplay.com)

ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL and Oracle Integration

MySQL integrates into the Oracle Environment



ORACLE®

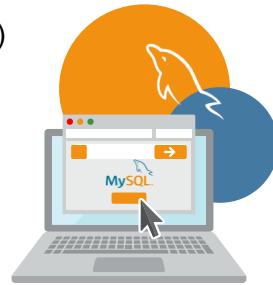
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

In addition to Oracle Enterprise Manager, MySQL is now integrated with nearly all relevant Oracle products.

Oracle wants to make it very easy for existing Oracle customers to integrate and manage MySQL in their current environment.

MySQL Websites

- <http://www.mysql.com> includes:
 - Product information
 - Services (Training, Certification, and Support)
 - White papers, webinars, and other resources
 - MySQL Enterprise Edition (trial version)
- <http://dev.mysql.com> includes:
 - Developer Zone (Forums, MySQL Engineering Blogs, and more)
 - Documentation
 - Downloads
- <https://github.com/mysql>
 - Source code for MySQL Server and other MySQL products



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Download MySQL Community Edition general availability (GA) and development releases from the <http://dev.mysql.com> website. This site also hosts the online documentation, which is an extremely valuable resource for both beginners and expert users of MySQL, and includes several example databases. Download trial versions of MySQL Enterprise Edition software, view detailed product information, and find out more about Oracle MySQL services at <http://www.mysql.com>.

Community Resources

- MySQL Community Slack
- MySQL Forums
- MySQL Newsletter (published monthly)
- Planet MySQL blogs
- Social media channels
 - Facebook
 - Twitter
 - LinkedIn
 - YouTube
- Physical and virtual events, including:
 - Developer days
 - MySQL Tech Tours
 - Webinars
- Bug tracking
- Worklogs
- GitHub repositories



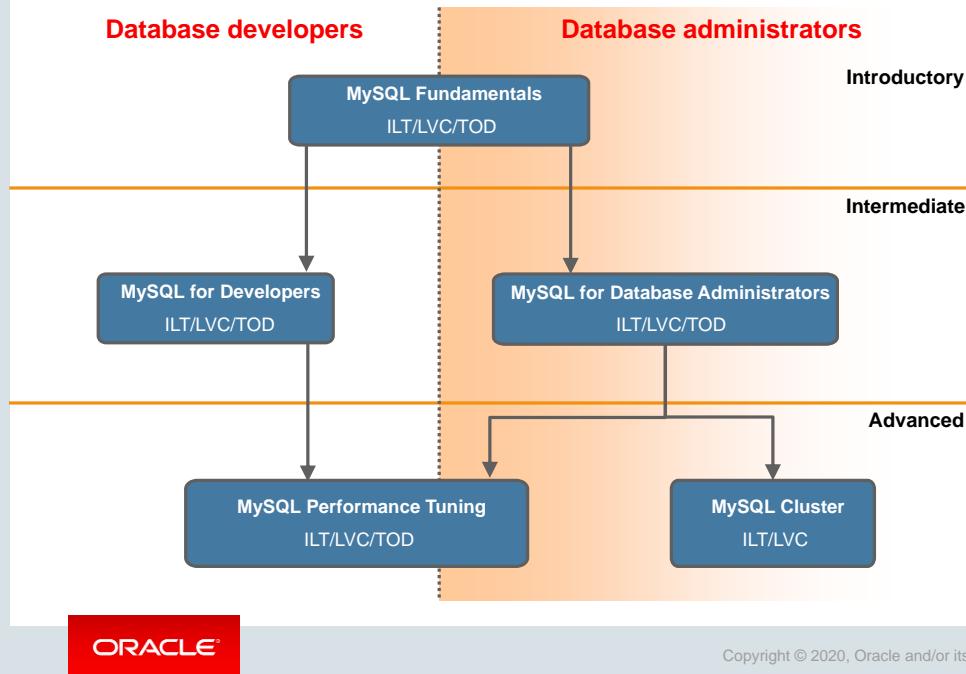
ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL has a very large and active community. Engage with your peers, report bugs, share information, and discover what is happening in the world of MySQL by using the following resources:

- MySQL Community Slack (<https://mysqlcommunity.slack.com/>)
- MySQL Forums (<http://forums.mysql.com>)
- MySQL Newsletter (<http://www.mysql.com/news-and-events/newsletter/>)
- Planet MySQL blogs (<http://planet.mysql.com>)
- Social media channels:
 - Facebook: <http://facebook.com/mysql>
 - Twitter: https://twitter.com/mysql_community and <http://twitter.com/MySQL>
 - LinkedIn: <https://www.linkedin.com/company/mysql>
 - YouTube: <https://www.youtube.com/user/MySQLChannel>
- Physical and virtual events (<http://www.mysql.com/news-and-events>)
- Bug tracking (<http://bugs.mysql.com>)
- Worklogs (<https://dev.mysql.com/worklog/>)
- GitHub repositories (<https://github.com/mysql>)

Oracle University: MySQL Training



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Training Formats

- **Instructor-Led Training (ILT):** Delivered in a classroom with instructor and students present at the same location and time
- **Live Virtual Class (LVC):** Delivered by using video and audio through a web-based delivery system (WebEx) in which geographically distributed instructors and students participate, interact, and collaborate in a virtual class environment
- **Training On Demand (TOD):** On-demand training that takes traditional classroom training (complete with all classroom content including lectures, white boarding, and practice videos) and makes it available in a video-based, online format so that you can start your customized training at your convenience

For full details about MySQL training options, go to <http://education.oracle.com/mysql>.

MySQL Certification

The Oracle Certification Program validates your expertise in the following areas:

- Oracle Certified Professional: MySQL Database Administrator
- Oracle Certified Professional: MySQL Developer

Take the examination at a local Pearson VUE test center:

- At a time suitable to you
- In over 175 countries



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

For full details of MySQL and other Oracle Certification options, visit:
<http://education.oracle.com/certification>.

Summary



In this lesson, you should have learned how to:

- Describe the course goals
- List MySQL products and professional services
- Access MySQL information and services from Oracle websites and community resources
- Find information about MySQL courses and certification options



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Practices

- 1-1: Course Environment



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

2

Installing and Upgrading MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Objectives



After completing this lesson, you should be able to:

- Install the MySQL server and client programs
- Identify the files and folders created during installation
- Perform the initial configuration of the MySQL server
- Start and stop MySQL
- Upgrade to MySQL 8.0



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

- Installing MySQL
 - Installed Files and Directories
 - Initial Configuration
 - Starting and Stopping MySQL
 - Upgrading MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Installation Sequence

1. Determine support for your platform.
 - See: <https://www.mysql.com/support/supportedplatforms/database.html>
2. Choose a distribution.
 - Pre-packaged binaries
 - Native formats, for example, RPMs for Linux, PKG for OS X, MySQL Installer for Windows
 - Generic formats, for example, Zip archives or compressed tar files
 - Source code
3. Download the distribution.
4. Install the distribution.
5. Perform any required post-installation configuration.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

In general, you should choose a binary distribution. Choose a source code distribution only if you want to see very recent new features and test new code.

Installing MySQL from Downloaded Packages

- After you download MySQL Linux packages, install them by using the following commands:
 - On RPM systems:

```
rpm -ivh packagename.rpm
```
 - On APT systems:

```
dpkg -i packagename.deb
```
- You must identify and resolve dependencies when you install packages.
 - Install any dependencies before you install MySQL packages.
 - For example, MySQL has a dependency on the `libaio` library. Data directory initialization and subsequent server startup steps will fail if this library is not installed.
 - Install MySQL packages in the correct order.
 - For example, install the `libs` package before installing `client` or `server`.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

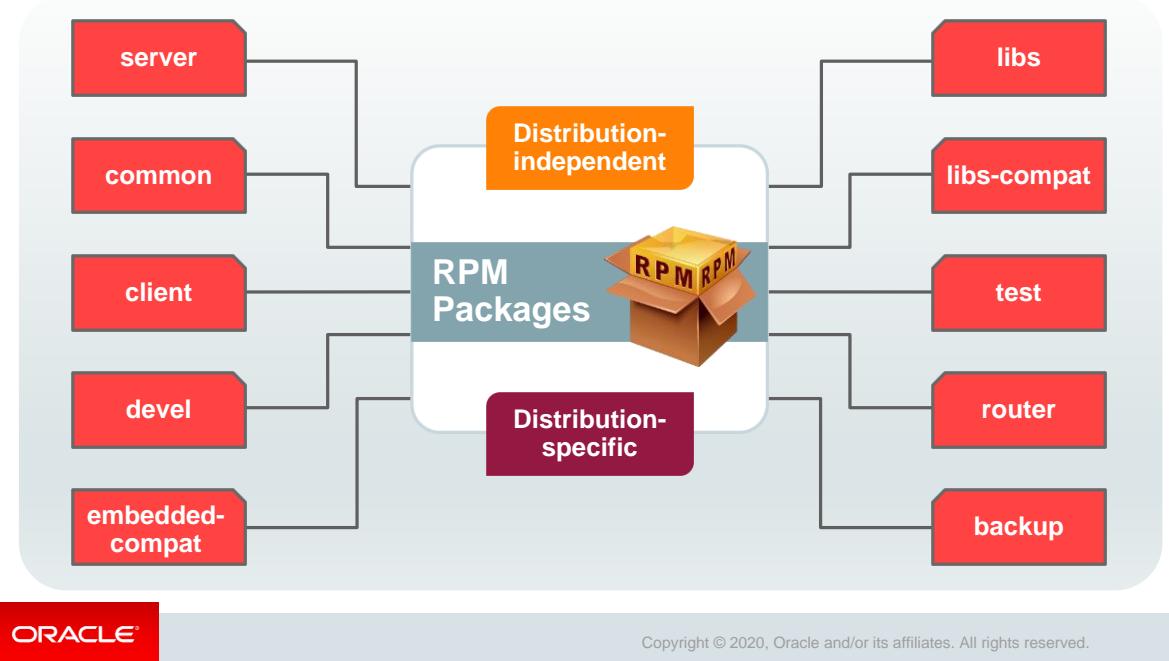
Installing Dependencies

On RPM machines that can reach repositories over the network or on a connected filesystem, you can resolve dependency problems by using your package manager.

If you are installing on a machine that cannot reach repositories, you must manually resolve any unmet dependencies that you identify while installing the MySQL packages. To do this, you must find the RPM or DEB files that contain the packages by searching the repositories appropriate to the Linux system on which you are installing.

Note: For information about using your package manager, see “Installing MySQL by Using a Package Manager” later in this lesson.

MySQL RPM Installation Files for Linux



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Available RPMs

Oracle provides two types of MySQL RPMs:

- **Distribution-independent:** Should work on all versions of Linux that support RPM packages and use glibc 2.12
- **Distribution-specific:** Intended for a targeted Linux platform, such as Oracle Linux 7

An RPM installation for MySQL is typically split into different packages. You can download these individually or bundled into a single archive.

- **Server:** Database server and related tools
- **Common:** Common files for server and client libraries
- **Client:** MySQL client applications and tools
- **Devel:** Development header files and libraries for MySQL database client applications
- **Embedded-compat:** MySQL server as an embedded library with compatibility for applications using version 18 of the library
- **Libs:** Shared libraries for MySQL database client applications
- **Libs-compat:** Shared compatibility libraries for previous MySQL installations
- **Test:** Test suite for the MySQL server
- **Router:** Lightweight middleware that provides transparent routing in InnoDB cluster
- **Backup:** MySQL Enterprise Backup (commercial only)

For a standard installation, you must install at least the common, server, and client packages.

MySQL RPM Installation Process

- The RPM installation performs the following tasks:
 - Extracts RPM files to their default locations
 - Registers SysV init or systemd startup scripts
 - Sets up the `mysql` user and group in the operating system
 - The MySQL server process runs as the `mysql` user.
- When you start the service for the first time using `service mysqld start` or `systemctl start mysqld`, MySQL:
 - Creates the data directory and the `my.cnf` file
 - These files are owned by the `mysql` user and group.
 - Creates the default `root@localhost` account
 - Sets up a random temporary password for the `root` account and writes that password to the error log file (`/var/log/mysqld.log`)
 - You must change the password before you can use MySQL.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL File Locations

The RPM installer sets `basedir` to `/usr` and does not give an option to install MySQL to any other location on the filesystem. This might cause problems if you want to install multiple versions of MySQL on the same host.

To install to another location, use the generic binaries. Note that the RPM installer also sets up the `mysql` user and group, the Linux services, and other settings that must be done manually when you install from generic binaries.

Oracle RPMs and Non-Oracle RPMs

Some Linux distributions provide their own MySQL RPM builds. The steps in the slide reflect the installation process for Oracle RPMs. Because this behavior is built into the RPM, other distributions might differ.

Starting the MySQL Service

The MySQL service does not start automatically. You must start it yourself before you can use MySQL:

- `service mysqld start`
- `systemctl start mysqld` on Linux distributions that run systemd.

On Linux versions that run systemd, the `service` command will be mapped to an equivalent `systemctl` command.

MySQL DEB Installation

- DEB packages are available for APT Linux systems, either individually or bundled.
- For a standard installation, install the following packages in the specified order.
 1. The database common files package
 2. The client package
 3. The client metapackage
 4. The server package
 5. The server metapackage

```
sudo dpkg -i mysql-{common,community-client,client,  
community-server,server}_*.deb
```

- When you install the Server package, `dpkg` provides a configuration interface:
 - Choose a password for the `root` account.
 - Indicate if you want to install the `test` database.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

DEB Pre-installation Step

MySQL 8.0 does not allow the variable `lower_case_table_names` to be changed after the server has initialized. DEB installation automatically initialized the MySQL server during the installation. If you want to enable the `lower_case_table_names` setting, you need to use the `debconf-set-selection` utility to configure the option before starting the installation.

- `shell> sudo debconf-set-selections <<< "mysql-server mysql-server/lowercase-table-names select Enabled"`

DEB and RPM Package: Differences

DEB package components are organized similarly to those in RPM packages. Differences include the following:

- There is no DEB source package. You can obtain the source code as a `.tar.gz` file directly.
- There is no equivalent `libs-compat` package. Its purpose is specific to RPM-based distributions.
- The `dpkg` installation process is interactive by default. When you install the package, it displays a Curses-based interface in which you set the options shown in the slide. You also see this interface when you use the `dpkg-preconfigure` command to set those configuration options before performing the installation, at which time the interface does not appear. If you have set the `DEBIAN_FRONTEND` environment variable to `noninteractive`, the Curses interface does not appear, and the `root` password remains blank.

Linux Distribution–Specific Repositories

Many Linux distributions maintain their own package repositories.

- Hosted on the web or on a local filesystem
- Accessed by using package managers
 - The repository website is specified in the package manager configuration on each host.
 - You can download individual packages manually.
- Contain software packages and their dependencies
- Supplemented by additional repositories that contain software not contained in the standard package repositories
 - You can add these additional repositories to the package manager configuration on each host.
- Example: <http://public-yum.oracle.com/> contains Oracle Linux packages.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Local Repositories

When you install Linux from a CD or DVD, the installation medium might contain a repository of selected packages that the installer uses during the installation process. You can install packages from such a repository without connecting to the Internet.

Installing MySQL by Using a Package Manager

- On RPM-based systems, including Oracle Linux, Red Hat, Fedora, and CentOS, use `yum install`.

- Example:

```
yum install mysql-community-server  
yum install mysql-workbench
```

- On APT-based systems, including Ubuntu and Debian, use `apt-get install`:

- Example:

```
apt-get install mysql-community-server  
apt-get install mysql-workbench
```

- Installing the `mysql-community-server` package also installs the packages for the components the server requires.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Dependencies

Package managers detect and install dependencies automatically, if they can reach their repositories over the network or from a filesystem. For example, the `mysql-community-server` package depends on several other packages, including:

- `mysql-community-client`: The client software
- `mysql-community-common`: The common error messages and character sets for both the client and server
- `mysql-community-libs`: The shared client libraries

If these dependencies do not exist on your system when you install `mysql-community-server`, the package manager finds and installs these packages and any dependencies they, in turn, might have.

Repository Maintainers

In some distributions, `mysql-server` is a virtual package that does not itself contain any software but, instead, has a specific version of `mysql-community-server` or some other software as a dependency. When you install such a virtual package, the package manager installs all of its dependencies including the package-defined version of MySQL.

Because this package is often maintained by the distribution maintainers and not Oracle, it does not reflect the latest changes in the MySQL software.

Instead, you can add a MySQL repository that contains packages that are created and maintained by Oracle's MySQL Release Engineering team. The following slides show how to do this.

Adding a Yum Repository

- Download the Yum repository RPM file from <http://dev.mysql.com/downloads/repo/yum/>.
 - Choose the correct RPM for your distribution.
 - Example: The “Red Hat Enterprise Linux 7 / Oracle Linux 7 (Architecture Independent), RPM Package” is called `mysql80-community-release-el7-3.noarch.rpm`.
- Install the file by using the `yum localinstall` command, for example:

```
yum localinstall mysql80-community-release-el7-3.noarch.rpm
```

 - The preceding command adds the MySQL Yum repository to the host’s Yum configuration.
- Enable or disable specific versions.
 - MySQL 8.0 is enabled by default. Other versions are disabled.
- Run `yum install packagename` to install a package from the new repository.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Repository Configuration

When you install the repository RPM file, it creates two files in `/etc/yum.repos.d/`:

- `mysql-community.repo`: Contains repository metadata for MySQL Community Edition
- `mysql-community-source.repo`: Contains repository metadata for the MySQL source code

The default configuration enables the most recent GA version of MySQL and the MySQL tools. Previous GA versions are disabled, as are development versions that are not yet generally available. You can edit these files to enable or disable specific available versions in the repository.

Configuring Yum Repository Versions

Enable or disable specific versions by editing the `/etc/yum.repos.d/mysql-community-repo` file:

- The latest Generally Available (GA) version is enabled by default.
- The following extract shows the MySQL 5.7 repository is disabled:

```
# Enable to use MySQL 5.7
[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

Change to `enabled=1` to enable the mysql57-community repository.

- To enable the MySQL 5.7 repository, change the value of the `enabled` setting so that it reads `enabled=1`.
- Use the same approach to enable development versions.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Installing MySQL from the Repository

When you install the repository RPM file, the `yum install` command automatically retrieves packages from the most suitable repository. The official MySQL repositories usually have more recent versions than the distribution's own packages, so `yum` preferentially installs the official packages because they represent the most recent version in its repositories.

Similarly, if you execute a `yum upgrade` command to upgrade all packages on your machine after installing the new repository, then the package manager automatically upgrades any previous version of MySQL that you had previously installed from its own repositories and replaces them with the new official version.

Adding an APT Repository

- Download the APT repository DEB file from <http://dev.mysql.com/downloads/repo/apt/>.
 - Supported distributions:
 - Debian version 9
 - Ubuntu LTS versions 16.04, and 18.04 and 18.10
- Install the file by using the `dpkg` command:

```
dpkg -i mysql-apt-config_0.8.12-1_all.deb
```

 - The preceding command adds the MySQL APT repository to the host's APT configuration.
- Configure the repository versions.
- Run `apt-get update` to refresh the repository metadata.
- Run `apt-get install packagename` to install a package from the new repository.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

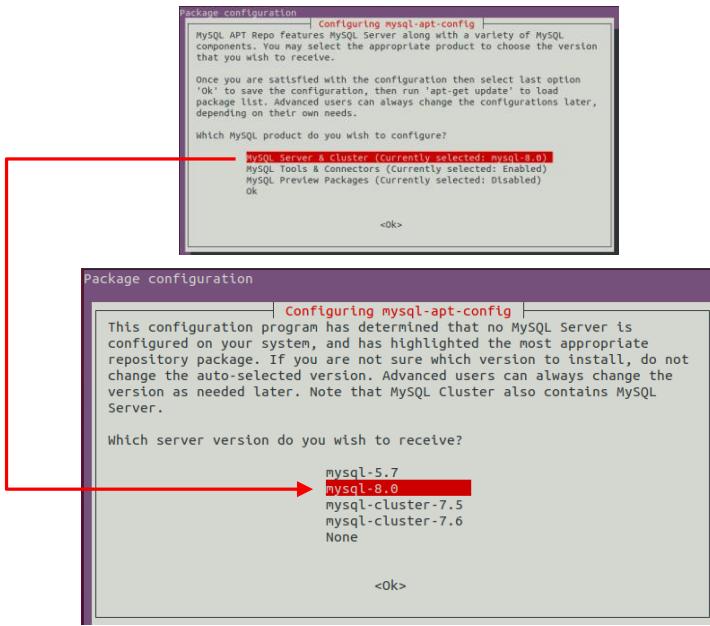
Repository Configuration

When you install the `mysql-apt-config` package, `dpkg` opens an interactive window from which you can choose the repository versions. This interface is shown in the following slide.

To reconfigure the repositories by using this interface, you can either reinstall the `mysql-apt-config` package or use the following command to reconfigure it:

```
dpkg-reconfigure mysql-apt-config
```

Configuring Repository Versions



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

dpkg Configuration Interface

The first page of the interface shows the products available in the repository configuration:

- MySQL Server and Cluster
- MySQL Tools and Connectors
- MySQL Preview Packages

It also provides an OK option that saves your changes and exits the interface.

When you select one of the products, the interface displays a selection of versions for that product. The example in the slide shows the `mysql-8.0` version of the Server product.

Manually Configuring the APT Repositories

Enable or disable specific repositories by editing
/etc/apt/sources.list.d/mysql.list:

- Performing the steps in the preceding slide results in the following configuration:

Lines that are
commented with a
“#” symbol indicate
disabled repositories.

```
### THIS FILE IS AUTOMATICALLY CONFIGURED ###
# You may comment out entries below, but any other
modifications may be lost.
# Use command 'dpkg-reconfigure mysql-apt-config' as root
for modifications.
deb http://repo.mysql.com/apt/ubuntu/ bionic mysql-apt-
config
deb http://repo.mysql.com/apt/ubuntu/ bionic mysql-8.0
deb http://repo.mysql.com/apt/ubuntu/ bionic mysql-tools
#deb http://repo.mysql.com/apt/ubuntu/ bionic mysql-tools-
preview
deb-src http://repo.mysql.com/apt/ubuntu/ bionic mysql-8.0
```

- Enable or disable a specific product repository by uncommenting or commenting its line, respectively.

ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

As suggested by the comment in the slide, manual changes to the mysql.list file are not retained if you reinstall or reconfigure the mysql-apt-config package; the package configuration interface prompts you for any changes you want to make to the configuration and then rewrites the file.

Installing MySQL on Windows

- MySQL Installer:
 - Is distributed as an `.msi` executable
 - Guides you through a configuration wizard to create the folders and configuration required to run MySQL
- Noinstall Archive:
 - Is distributed as a `.zip` file
 - Must be unpacked and moved to the desired installation location
 - Must be manually configured to create the folders and configuration required to run MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

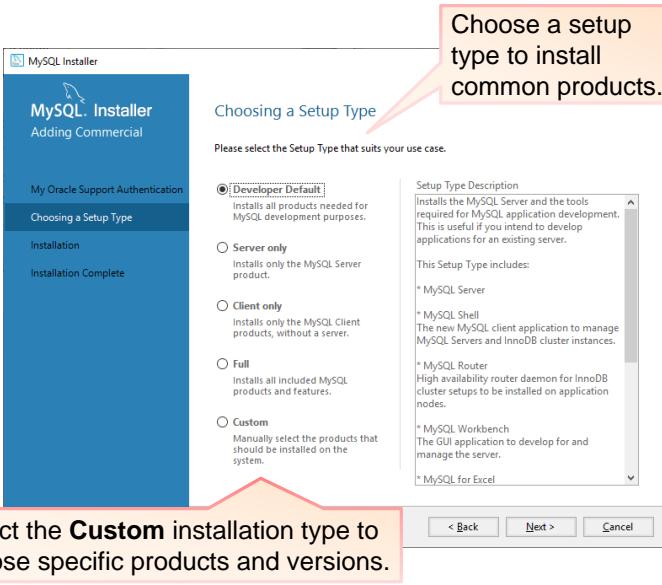
Included Products

The Windows distributions contain the MySQL database server, along with the following products:

- MySQL Workbench
- MySQL Notifier
- MySQL for Excel
- MySQL Enterprise Backup (commercial only)
- MySQL for Visual Studio
- MySQL Shell
- MySQL Router
- MySQL Connectors (.Net / Python / ODBC / Java / C++)
- Samples, examples, and documentation

Installing on Windows: MySQL Installer

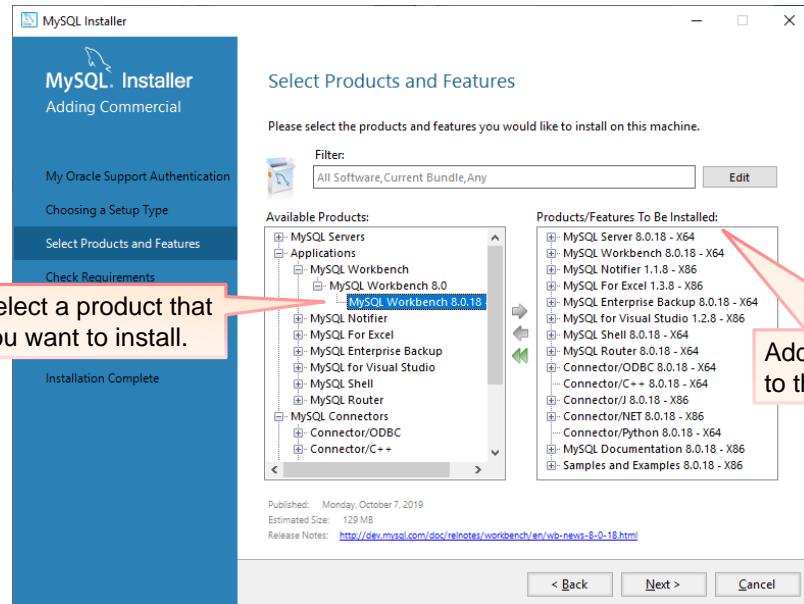
- Download the Full or Web installers:
 - **Full:** Contains all installation files in the downloaded file
 - **Web:** Contains only the installer
 - The installer downloads additional required files based on your selections during the installation process.
- Follow the Installer screens to select products and complete the installation.
 - License agreement
 - Setup installation type
 - Installation process
 - Post-install configuration



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

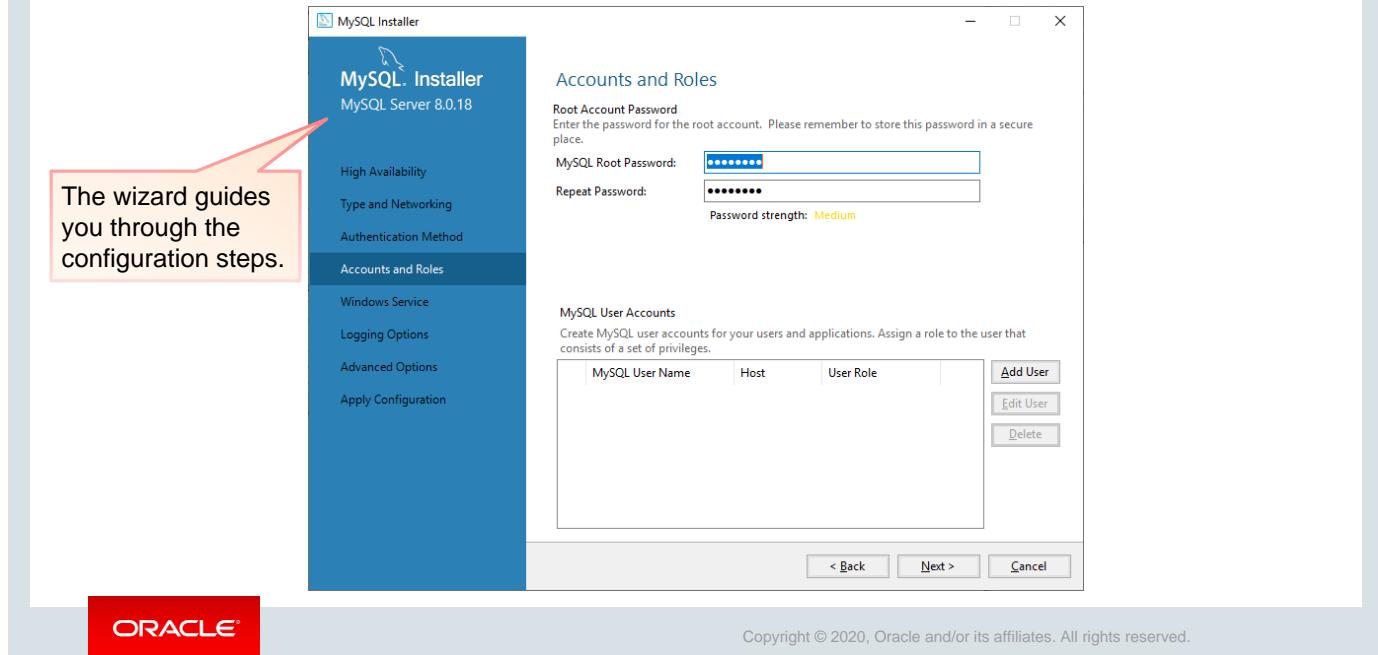
Installing on Windows: Selecting Products and Features



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Installing on Windows: Product Configuration



The wizard guides you through the following configuration steps:

- High Availability
- Type and Networking
- Authentication Method
- Accounts and Roles
- Windows Service
- Logging Options
- Advanced Options

Installing MySQL as a Windows Service

- With MySQL Installer:
 - Use the provided service name or select an alternative.
- At the command line after installation:
 - Install the service manually:

```
mysqld.exe --install servicename --defaults-file="C:\my.ini"
```
 - Remove an installed service:

```
mysqld.exe --remove servicename
```
- View the installed services by using the Services control panel application.
 - Launch from the command line:

```
services.msc
```
- Set services to start automatically or manually and provide a Windows account with which to start the service.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Windows Services

Each Windows service controls a program so that it can start and stop it. You must ensure that the program's path is available to the service manager when you create the service. You might need to provide the full path to the `mysqld.exe` executable.

Example

```
C:\mysql80\bin\mysqld.exe --install servicename ...
```

Instead of using the `--install` option, you can use the Windows `sc` command-line program to install MySQL as a Windows service.

Example

```
sc create servicename start=auto DisplayName=MySQL  
    binPath=C:\mysql80\bin\mysqld.exe
```

If you encounter problems in starting the service because you have spaces in the MySQL installation path (for example, if you use the default MySQL Installer path, which includes the "MySQL Server 8.0" text), then install the service by using the short path name or use double quotation marks around the `binPath` parameter.

Installing MySQL from Source

Build MySQL from the source code when you need to:

- Configure compiled-in options
 - Examples:
 - Disabling unused features on production servers with well-understood use cases to maximize performance
 - Enabling additional debugging features
- Run MySQL on a platform for which there are no precompiled binaries
- Add your own modifications (or community patches) to MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Compiling MySQL

Compile MySQL from source code if you require a feature that might not be available in a precompiled distribution (such as full debugging support). To produce a server that uses less memory when it runs, you may need to disable a feature that is not needed.

For example, you may need to disable optional storage engines or compile only those character sets that the application needs.

The requirements for compiling MySQL are similar to those when you compile any C++ program:

- CMake
- make
- GCC 5.3 (on Linux) or another ANSI C++ compiler

On Oracle Linux 7, you need to enable the software collections to install the required tools to compile MySQL software.

There are further requirements depending on the type of compilation you use and where you get the source code.

Note: See <https://dev.mysql.com/doc/mysql/en/source-installation.html> for a full discussion on installing MySQL from source.

Installing MySQL from Binary Archive

If you do not install from a package manager, you must perform some configuration steps manually.

1. Create the `mysql` user and group.

```
# groupadd mysql  
# useradd -r -g mysql -s /bin/false mysql
```

2. Extract the archive to a suitable directory while logged in as `mysql`.

- Alternatively, change the ownership of the extracted archive to `mysql` after extracting it.

```
# mkdir /usr/local/mysql-8.0.x/  
# chown mysql:mysql /usr/local/mysql-8.0.x/  
# cd /usr/local/mysql-8.0.x/  
# tar xf ~/mysql-8.0.x-linux-glibc2.12-x86_64.tar.gz
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

mysql User

You can use a different operating system username and group name instead of `mysql`. For simplicity, these instructions follow the same convention as that used in the installation packages, each of which creates a `mysql` user and group during the installation process.

If you use a different username and group name, ensure that the MySQL server process runs with that username and that the user has ownership of the MySQL data, log, and PID files.

Although it is possible to run the `mysqld` process as the `root` user, this is not recommended. As with any other operating system service, MySQL can perform only those operations for which its user is authorized. If such a service (or an application that communicates with it) is compromised, an attacker is limited by the restrictions placed on that user.

Installation directory

You can extract the archive to any directory for which you have write permission. MySQL server uses the `basedir` option to identify the location of the installation directory.

Installing MySQL from Binary Archive

3. Create the initial configuration file.

- Copy `my-default.cnf` to `/etc/my.cnf`.
- Edit the `datadir` setting to point to the data directory.
- Edit the `basedir` setting to point to the installation directory.
- Edit any other required settings:
 - Log file settings
 - TCP port

4. Initialize the data directory and note the generated temporary password:

```
# bin/mysqld --initialize --user=mysql
```

5. Start the MySQL Server.

```
# bin/mysqld_safe --user=mysql &
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The generated temporary password is expired and is shown in a message written to the error log file or console.

You may use `mysqld --initialize-insecure --user=mysql` to skip the automatic generation of a temporary password for the root user account.

Installing MySQL from Binary Archive

6. Connect to the MySQL server using the temporary password.

```
# mysql -u root -p  
Enter password: (enter the random password)  
...  
mysql>
```

7. Change the `root` user password.

```
mysql> ALTER USER USER() IDENTIFIED BY 'new password';
```

8. Optionally, populate time zone tables.

- Example:

```
# mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root mysql -p
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Time Zone Tables

When you create a new database directory, MySQL creates time zone tables but does not populate them. You can populate them by doing either of the following:

- Execute the `mysql_tzinfo_to_sql` utility to create `INSERT` statements based on the contents of your system's time zone database. Use this option if your system has `zoneinfo` files so that MySQL has the same time zone information as other system tools.
- Download a script containing `INSERT` statements from <http://dev.mysql.com/downloads/timezones.html>. Use this option if your system does not already have time zone information.

MySQL uses time zone information to handle zone-sensitive functions and data types such as `NOW()` and `TIMESTAMP` and to calculate illegal dates and times such as those that occur when transitioning to daylight savings time.

Deploying MySQL Server with Docker

You need to install the Docker engine and client on your host machine first.

1. Download a MySQL Server Docker image.

- Using the MySQL Community Edition image

- Log in to Docker Hub and pull the image using the `docker` command:

```
# docker pull mysql/mysql-server
```

- Using the MySQL Enterprise Edition image

- Log in to My Oracle Support and download the MySQL Commercial Server Docker Image

- Load the image using the `docker` command:

```
# docker load -i mysql-enterprise-server-version.tar
```

2. List all the downloaded images.

```
# docker images
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Deploying MySQL Server with Docker

3. Create a MySQL Server Docker container:

```
# docker run --name=mysql1 -d mysql/mysql-server
```

4. View the MySQL server logs:

```
# docker logs mysql1
```

- Retrieve the generated temporary password:

```
# docker logs mysql1 2>&1 | grep GENERATED
```

5. Connect to the MySQL server from within the container:

```
# docker exec -it mysql1 mysql -uroot -p
```

- Enter the temporary password
- Change the root user password:

```
mysql> ALTER USER USER() IDENTIFIED BY 'new password';
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

`docker run --name=container_name -d image_name:tag`

Quiz



For which installation method must you create the data directory manually?

- a. Binary Archive installation for Linux
- b. DEB installation for Linux
- c. MySQL Installer for Windows
- d. RPM installation for Linux



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Answer: a

Topics

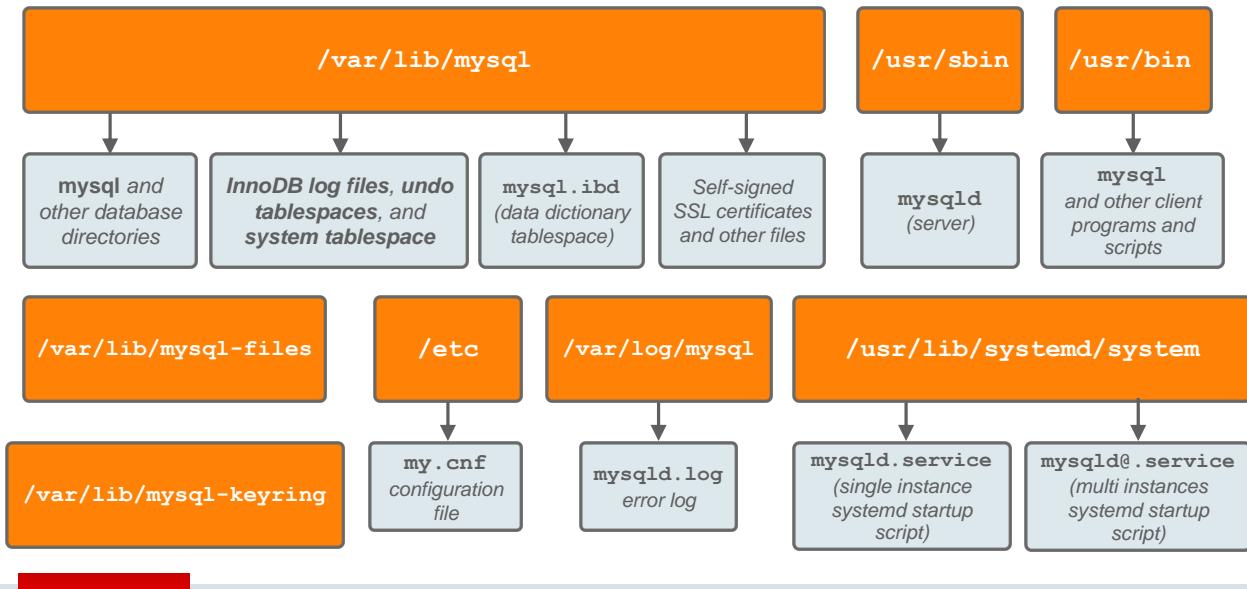
- Installing MySQL
- **Installed Files and Directories**
- Initial Configuration
- Starting and Stopping MySQL
- Upgrading MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Linux MySQL Server Installation Directories



ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The slide shows the main directory locations and files in a standard installation. These locations are used when you install from RPM and DEB packages.

Data Directory

- `/var/lib/mysql` is where the server stores databases. This directory is configured when you (or the installation process) run `mysqld --initialize`. The InnoDB log files, undo tablespaces, and system tablespace are in this directory. It also includes a subdirectory for each database. This includes the `mysql` directory, which contains system tables including the grant tables. MySQL 8.0 stores the data dictionary in `mysql.ibd` tablespace using InnoDB storage engine.

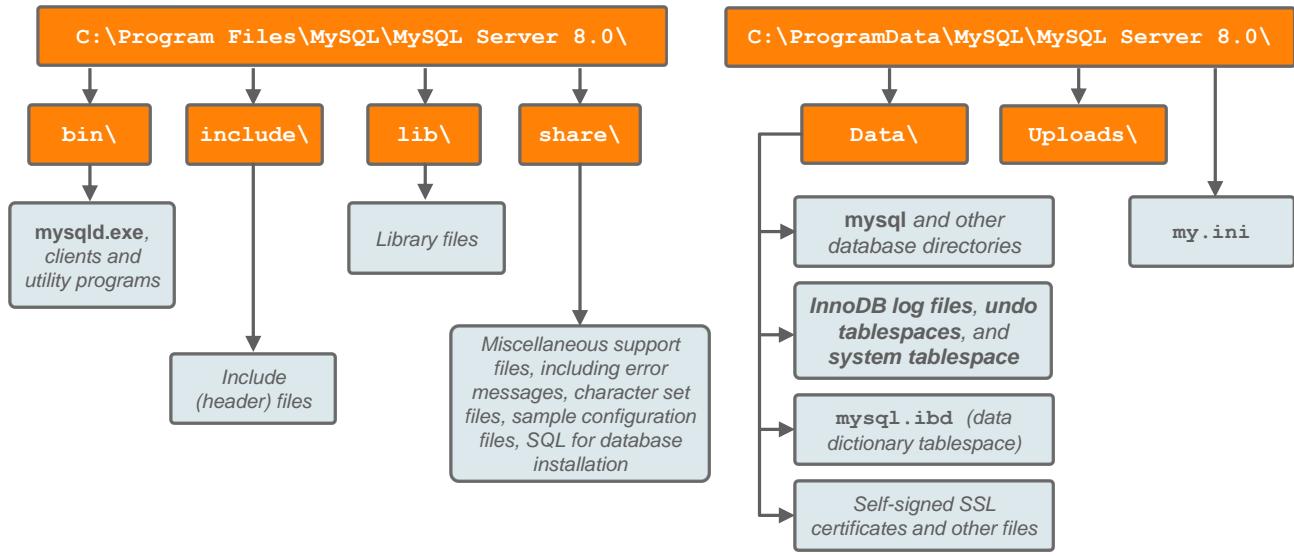
Base Directory

- `/usr/sbin` contains the main server executable, `mysqld`.
- `/usr/bin` contains client programs and scripts such as `mysql`, `my_print_defaults`, `mysqladmin`, `mysqlcheck`, and `mysqlimport`.

Other Directories

- `/var/lib/mysql-files` is configured in the `secure_file_priv` variable for storing import and export data files.
- `/var/lib/mysql-keyring` is allocated for storing keyring files.
- `/etc` and `/var/log` are standard Linux directories for configuration files and log files. The `/etc/my.cnf` file is read by the MySQL server process (`mysqld`).
- The systemd startup script is stored in the default systemd directory.

Windows MySQL Server Installation Directory



ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

By default, MySQL is installed in the `C:\Program Files\MySQL\MySQL Server version` directory. For example, the installer places MySQL 8.0 in the `C:\Program Files\MySQL\MySQL Server 8.0` directory.

Program Files Directory

`bin\` contains the MySQL server and client programs. Windows MySQL distributions include multiple programs, which can be found in the `bin` directory:

- **`mysqld.exe`**: The server executable
- Client programs, such as `mysql.exe` and `mysqladmin.exe` (listed in the slide titled "Command-Line Client Programs")

Other directories are used for include files, library files, and support files.

ProgramData Directory

`Data\` is where the server stores databases and log files. This directory is preconfigured and ready to use. For example, this directory includes a `mysql` subdirectory (contains the grant tables).

`Uploads\` is configured in the `secure_file_priv` variable for storing import and export data files.

Configuration File

The `my.ini` configuration option file specifies where the installation directory is located, as well as other optional settings.

MySQL Programs

- Server programs:
 - The `mysqld` server process
 - Service utilities, launcher programs
- Installation programs:
 - Programs that perform part of the initial installation configuration process
- Utility programs:
 - MySQL programs that perform some function without connecting to the server
- Client programs:
 - Programs that connect to the server



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The following slides list some of the programs that are installed when you install MySQL on your computer. Some of the programs are covered in more detail later in this course when you learn about the tasks that you can perform with those programs.

mysqld: MySQL Server Process

- Launched automatically by one of the server helper programs
 - Including operating system startup scripts
- Launched manually to debug the MySQL server configuration
 - The error messages go to the terminal by default rather than to the error log.
 - Example:

```
$ mysqld --user=mysql --datadir=/var/lib/mysql  
--socket=/tmp/mysql.sock
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

mysqld Command-Line Options

The options shown in the slide are some of the many options you can use to configure an instance of MySQL server.

Note: Command-line options are covered in more detail in the lesson titled “Configuring MySQL.”

Installation Programs

- **mysql_secure_installation:**
 - Security program that enables initial secure configuration
- **mysql_tzinfo_to_sql:**
 - Utility that creates a SQL script containing the host's time zone information
- **mysql_upgrade:**
 - Program that verifies database contents and ensures that they are compatible with the current version of MySQL
 - Deprecated since MySQL Server 8.0.16 where the tasks are performed automatically when the server starts up



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

`mysql_secure_installation` enables you to improve the security of your MySQL installation in the following ways:

- You can set a password for root accounts.
- You can remove root accounts that are accessible from outside the local host.
- You can remove anonymous-user accounts.
- You can remove the `test` database (which by default can be accessed by all users, even anonymous users) and privileges that permit anyone to access databases with names that start with `test_`.

MySQL 8.0 installations have handled most of these issues by default. Therefore, it may not be necessary to run `mysql_secure_installation`.

Utility Programs

- **mysql_config_editor**
 - Manages login paths to simplify how you connect command-line clients to the MySQL server
- **mysqlbinlog**
 - Reads and replays the contents of the binary log
- **mysqldumpslow**
 - Reads and summarizes the contents of the slow query log
- **mysql_ssl_rsa_setup**
 - Creates TLS keys and certificates
- **ibd2sdi**
 - Extracts serialized dictionary information (SDI) from InnoDB tablespace files



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

`mysql_config_editor`

Use `mysql_config_editor` to create encrypted option files.

- Store user, password, and host options in a dedicated option file:
 - `.mylogin.cnf` in the current user's home directory
 - To specify an alternative file name, set the `MYSQL_TEST_LOGIN_FILE` environment variable.
- The `.mylogin.cnf` file contains login paths.
 - They are similar to option groups.
 - Each login path contains authentication information for a single identity.
 - Clients refer to a login path with the `--login-path` (or `-L`) command-line option:

```
# mysql --login-path=admin
```
 - Protect the file from being read by other users. Anyone who can read the file can use the credentials and is able to obtain the plain text passwords.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Obscuring Saved Passwords

Specifying a password on the command line in the form `mysql -uroot -ppassword` is not recommended. For convenience, you could put a password in a `[client]` option group, but the password is stored in plain text, easily visible to anyone with read access to the option file.

The `mysql_config_editor` utility enables you to store authentication credentials in an encrypted login file named `.mylogin.cnf`. The file location is the current user's home directory on Linux and UNIX and the `%APPDATA%\MySQL` directory on Windows. The file can be read later by MySQL client programs to obtain authentication credentials for connecting to a MySQL server. The encryption method is reversible, so you cannot presume that the credentials are secure against anyone who has read privileges to the file. Rather, the feature makes it easier for you to avoid using plain text credentials either at the command line or in MySQL configuration files.

.mylogin.cnf Format

- The decrypted .mylogin.cnf file consists of option groups.
 - Similar to other option files
- Each option group in .mylogin.cnf is a login path.
 - A set of values indicating the server host and the credentials for authenticating with that server
 - Permits only a limited set of options (user, password, and host)
- Example:

```
[admin]
user = root
password = oracle
host = 127.0.0.1
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

If you try to read the .mylogin.cnf file in a text editor, it appears in its encrypted form. You can view the login paths by using the print command, shown in the next slide.

Note: The my_print_defaults utility with the -s option can show the password in plain text.

Login Paths

- To create a login path:

```
mysql_config_editor set  
  --login-path=login-path --user=username  
  --password --host=hostname
```

- To view a single login path in clear text:

```
mysql_config_editor print  
  --login-path=login-path
```

- To view all login paths in clear text:

```
mysql_config_editor print --all
```

- To remove a login path:

```
mysql_config_editor remove  
  --login-path=login-path
```

- The default login path name is [client]. It is read by all standard clients.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

If you invoke `mysql_config_editor` without using the `--login-path` option, it uses the `[client]` login path. This login path is used by all standard clients by default.

For example, the following command creates a `[client]` login path used by all standard clients:

```
# mysql_config_editor set --user=root --password  
Enter password: oracle
```

Invoking a standard client with no further command-line arguments or option files causes it to read the `[client]` login path in the `.mylogin.cnf` file, along with `[client]` option groups in any option files. For example, the following output shows the result of invoking the `mysql` client with no options, having executed the preceding command:

```
# mysql  
Welcome to the MySQL monitor. Commands end with ; or \g.  
...
```

Command-Line Client Programs

- mysql: MySQL command-line client
- mysqladmin: Utility for monitoring, administering, and shutting down MySQL
- mysqldump/mysqlpump: Backup utilities that create SQL scripts to restore the structure and contents of databases
- mysqlimport: Utility for importing the contents of delimited data files
- mysqlslap: Load emulation client
- mysqlshow: Utility for displaying database object metadata
- mysqlcheck: Utility for checking and optimizing tables
- mysqlsh: MySQL Shell is an advanced command-line client and code editor for MySQL Server.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL Shell has to be installed separately.

MySQL Workbench is a graphical client program that is installed separately and does not use the same command-line options as the clients shown in the slide.

Launching Command-Line Client Programs

- Clients support many common options:
 - Authentication and connectivity options
 - Examples: `--user`, `--password`, `--host`, `--socket`
- Some options are specific to each client:
 - `mysqladmin` example options:
 - `--relative`
 - `--sleep`
 - `mysqlcheck` example options:
 - `--analyze`
 - `--check`
 - `--check-upgrade`



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

- Installing MySQL
- Installed Files and Directories
- **Initial Configuration**
- Starting and Stopping MySQL
- Upgrading MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Configuring Mandatory Access Control

Some platforms implement Mandatory Access Control (MAC) systems that:

- Prevent services from accessing unauthorized resources, such as files or ports
 - The default configuration includes default file and port access for common services including MySQL.
- Can be configured to enable specific nondefault access
- Can be disabled entirely
- Include:
 - Linux MAC systems:
 - SELinux: Oracle Linux, Red Hat Enterprise Linux, Fedora, CentOS
 - AppArmor: SUSE, Ubuntu
 - Oracle Solaris Extended Policy



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Default MAC Configuration

Because MySQL is such a commonly used service on many platforms, common MAC systems, by default, enable the `mysqld` server process to access the default MySQL TCP port and default data directory and log locations. In general, you do not need to modify MAC configuration unless you want to enable a nondefault configuration, for example, if you change the data directory or TCP port number.

SELinux Example

- Add new ports mapping.
 - Set a range of TCP ports: 33060 (X Protocol), 33061 (Group Replication), and 33062 (admin port) to the `mysqld_port_t` type so that MySQL can use them:
- Add a new file context type mapping.
 - Set the `/datadir` directory and its contents to the `mysql_db_t` type so that MySQL can access them:

```
# semanage port -a -t mysqld_port_t -p tcp 33060-33062  
  
# semanage fcontext -a -t mysql_db_t "/datadir(/.*)?"  
# restorecon -Rv /datadir  
restorecon reset /datadir context  
unconfined_u:object_r:default_t:s0-  
>unconfined_u:object_r:mysql_db_t:s0
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Disabling SELinux

You can temporarily disable the blocking effect of SELinux by setting it to *permissive* mode. To do this, execute the following command:

```
setenforce 0
```

In permissive mode, SELinux logs unauthorized access to its log file (by default, in `/var/log/audit/audit.log`), but does not prevent such access. Use permissive mode to troubleshoot access issues that you suspect might be caused by SELinux configuration. When you reboot the system or execute the `setenforce 1` command, SELinux resumes enforcing its policies.

Disable SELinux entirely by changing the `SELinux` setting from `enforcing` to `disabled` in the `/etc/selinux/config` file, as in the following example:

```
SELINUX=disabled
```

AppArmor: Example

- Edit the `/etc/apparmor.d/local/usr.sbin.mysql` file to change the MySQL policy on this machine.

- Set the `/datadir` directory and its contents to be accessible.

```
# /datadir/** rwk,
```

- Reload AppArmor profiles.

```
# service apparmor reload
* Reloading AppArmor profiles
```

- Set the MySQL policy to *complain* mode.

```
# aa-complain /usr/sbin/mysql
Setting /usr/sbin/mysql to complain mode.
```

- Set the MySQL policy to *enforce* mode.

```
# aa-enforce /usr/sbin/mysql
Setting /usr/sbin/mysql to enforce mode.
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Complain and Enforce Modes

AppArmor, like other MAC systems, prevents unauthorized access by default. This is called *enforce* mode. Execute the `aa-complain` command to change the policy for a specific executable to *complain* mode when you want that policy to log unauthorized access but not prevent it.

Disabling AppArmor

AppArmor runs as a service. To disable it, disable the `apparmor` service or startup scripts by using your operating system's service management features. For example, on systems that use `systemd`, execute the following command:

```
systemctl disable apparmor.service
```

Changing the root Password

- Change the root password after installation:
 - RPM installations create a `root` account with an initial random expired password, which you must change before using.
 - Written to the server log (usually in `/var/log/mysql/mysqld.log`)
 - Interactive DEB installations prompt for an initial `root` password that does not need to be changed.
 - Installations provided by other maintainers might create an initial root password in some other way.
 - An initial random expired password is generated when MySQL is initialized using `mysqld --initialize`. Source and archive installations usually use this method.
- Log in to MySQL with the initial password and use the `ALTER USER` statement to change it:

```
# mysql --user=root --password
Enter password: drVXqk9)nn?t
mysql> ALTER USER USER() IDENTIFIED BY 'neW%P4ss';
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Using mysqladmin to Change the root Password

Use the `mysqladmin` client utility with the `password` command to change the password.

- If you do not provide an argument to the `password` command, `mysqladmin` prompts for the new password:

```
# mysqladmin --user=root --password password
Enter password: drVXqk9)nn?t
New password: neW%P4ss
Confirm new password: neW%P4ss
```

- If you provide either the old or the new password on the command line, `mysqladmin` displays a warning:

```
# mysqladmin --user=root --password password 'neW%P4ss'
Enter password: drVXqk9)nn?t
mysqladmin: [Warning] Using a password on the command line interface can be
insecure.
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Using Login Paths

You can use login paths with many MySQL clients including `mysqladmin`. The following example shows how to set a new password by using the default `[client]` login path:

```
# mysqladmin password
New password: neW%P4ss
Confirm new password: neW%P4ss
```

Note that after changing the user password with this command, the client login path no longer contains the correct user password, so you must execute `mysql_config_editor` to set the new user password.

Quiz



Many installations of MySQL 8.0 are “secure by default” and create a random root password at installation time. Which two of the following are valid ways to change that password?

- a. Specify a new password in `/etc/my.cnf` (Linux) or `my.ini` (Windows).
- b. Issue a `mysqladmin ... password` command at the command line to change the initial password.
- c. Launch MySQL with the `mysqld --initialize` command.
- d. Log in to MySQL with the initial password and issue an `ALTER USER` statement to change it.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Answer: b, d

Topics

- Installing MySQL
- Installed Files and Directories
- Initial Configuration
- Starting and Stopping MySQL
- Upgrading MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Starting and Stopping MySQL

- Methods of starting MySQL:
 - Run the `mysqld` binary directly.
 - Run `mysqld_safe`.
 - Use the Linux service manager
 - Run `service mysqld start` on SysVInit systems.
 - Run `systemctl start mysqld` on systemd.
- Methods of stopping MySQL:
 - Kill the `mysqld` binary with the `SIGTERM` signal (-15).
 - Kill `mysqld_safe` first if it is running.
 - Use the Linux service manager
 - Run `service mysqld stop` on SysVInit systems.
 - Run `systemctl stop mysqld` on systemd.
 - `mysqladmin shutdown`
 - `SQL SHUTDOWN statement`



ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Using `kill`

The `service mysqld stop` command (or `systemctl stop mysqld` on systemd distributions) sends the UNIX `SIGTERM` signal (`kill -15`) to the `mysqld` process, which the process interprets as an instruction to shut down the server. When you use `mysqladmin shutdown`, it uses a server protocol command to send the “shut down” message to the server. In both cases, the server process receives the message and performs the same orderly shutdown procedure.

If you send the `SIGKILL` signal (`kill -9`), `mysqld` stops immediately without an orderly shutdown, as if you had pulled the power cord on the machine. This is likely to cause data corruption. If the `mysqld_safe` helper program is running, it detects this sudden shutdown as a crash and restarts `mysqld`.

SQL RESTART statement

MySQL 8.0 introduces the `RESTART` SQL statement, which stops and restarts the MySQL server. `RESTART` requires a monitoring process such as `systemd`, `mysqld_safe`, or Windows Service to start the MySQL server after it has been shut down with exit code 16.

Stopping MySQL with mysqladmin

- Using the [client] login path:

```
mysqladmin shutdown
```

- Using the [admin] login path:

```
mysqladmin --login-path=admin shutdown
```

- Providing credentials and server connections at the command line:

```
mysqladmin -u root -p -h dbhost -P 3306 shutdown
```

```
Enter password: password
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

SHUTDOWN Privilege

Some `mysqladmin` commands are available only to MySQL accounts that have the required privileges. For example, to shut down the server, you must connect to it by using an administrative account such as `root` that has the `SHUTDOWN` privilege.

MySQL Service Files

- On SysVInit systems, copy the `mysql.server` script to `/etc/init.d/mysqld`.
 - Some package installers create this file automatically.
 - Call it with `start`, `stop`, or `restart` options.
 - Examples:

```
service mysqld start
/etc/init.d/mysqld restart
support-files/mysql.server stop
```

- When you install MySQL from an RPM package built for systemd distributions or if you have configured systemd manually, use the `systemctl` command:

```
systemctl start mysqld
systemctl stop mysqld
```

- This uses the `mysqld.service` file, which is a systemd service unit configuration file.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

SysVInit, Upstart, and `systemd` Distributions

SysVInit is a standard method for initializing UNIX services. Many Linux systems have used this method. In recent years, the Upstart process replaced SysVInit on many distributions, emulating its features. The `mysql.server` script operates the same way on SysVInit and Upstart systems.

The latest versions of some Linux distributions (including those supported by MySQL) use `systemd` as the replacement initialization process. If you install MySQL from an RPM or APT binary, the service is automatically configured. If you install MySQL by using generic Linux binaries, you must configure the `systemd` service yourself.

Note: You configure the `mysqld` service in `systemd` in the practices titled "Configuring the MySQL Service."

Starting and Stopping MySQL on Windows

- Run the server process directly.
 - `mysqld.exe`
- If you have installed MySQL as a Windows service:
 - An automatic service starts when Windows starts
 - Start and stop services manually from the Services control panel application
 - Launched from the Start Menu or with the `services.msc` command
 - Start and stop services manually from the command line with the `net` or `sc` commands:

```
net start servicename  
sc start servicename  
net stop servicename  
sc stop servicename
```



ORACLE®

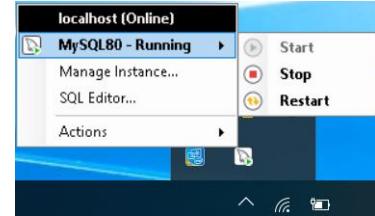
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Manual and Automatic Service Startup

To start and stop the service by using the Windows Services control panel application, select the MySQL service in the list of services and then click the Start or Stop link. You can configure manual or automatic startup in the Services application: A manual service starts only when you start it directly, whereas an automatic service starts when Windows starts.

Starting and Stopping MySQL on Windows: MySQL Notifier

- Installed by MySQL Installer
- Automatically registers MySQL services on the local machine
- Enables registration of remote MySQL services
- Displays the running status of registered servers
 - Displayed in the system tray
 - Optionally, notifies when a registered server changes status or when MySQL Notifier detects a new local MySQL service
- Enables starting, restarting, and stopping registered servers
- Launches installed MySQL applications:
 - MySQL Workbench
 - MySQL Utilities
 - MySQL Installer



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

- Installing MySQL
- Installed Files and Directories
- Initial Configuration
- Starting and Stopping MySQL
- Upgrading MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Upgrading MySQL

- The easiest upgrade is between minor versions within the same series.
 - Example: Upgrading from 8.0.xx to 8.0.yy
- You can upgrade from MySQL 5.7 to 8.0
 - Only between General Availability (GA) releases, MySQL 5.7.9, or later
 - Oracle recommends upgrading to the latest 5.7 GA release before upgrading to MySQL 8.0.
- Be aware of differences between the versions so that you can choose the correct upgrade method (in-place or logical) and avoid compatibility problems.
- Even if you are performing an in-place upgrade, you should back up your data beforehand.
 - This enables you to roll back the upgrade if you encounter problems.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Downgrading

Downgrading from MySQL 8.0 to MySQL 5.7, or from a MySQL 8.0 release to a previous MySQL 8.0 release, is not supported. The only supported alternative is to restore a backup taken *before* upgrading. It is therefore imperative that you back up your data before starting the upgrade process.

Reading Release Notes

Before you upgrade, view the changes between your existing version and the target version:

- For MySQL 8.0, use the following URL:
 - <https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html>
- For previous versions, replace 8.0 in the URL. For example, MySQL 5.7:
 - <https://dev.mysql.com/doc/refman/5.7/en/upgrading-from-previous-series.html>
- You may want to view the release notes for the changes between the minor versions.
 - <https://dev.mysql.com/doc/relnotes/mysql/8.0/en/index.html>
- Look for the following headings, which denote changes that you might need to handle before you upgrade:
 - Known Issue
 - Incompatible Change
 - Functionality Added or Changed
 - Important Change



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Searching for Specific Changes

If you know that an option or some other feature has changed, but you do not know the version in which it changed, then you can use the search feature on the documentation page.

Alternatively, you can search for specific release notes on one of the common search engines by including the `site:` modifier to limit the search to the MySQL release notes webpage. For example, to search for the release notes for the `authentication_string` column, enter the following search criteria in one of the common search engines such as Google or Bing:

```
site:dev.mysql.com/doc/relnotes authentication_string
```

MySQL Shell Upgrade Checker Utility

- The `util.checkForServerUpgrade()` function:
 - Is available in MySQL Shell
 - Enables you to verify whether MySQL server instances are ready for upgrade
 - Checks for compatibility errors and issues for upgrade to MySQL 8.0
 - Supports only MySQL Server 5.7 and 8.0 General Availability (GA) releases
 - Checks the configuration file (`my.cnf` or `my.ini`) if you provide the file path
- The version of MySQL Shell must be the same or later than the version of MySQL Server to which you are upgrading.
- The following example checks a MySQL server for upgrade to release 8.0.18:

```
mysqlsh -- util checkForServerUpgrade user@localhost:3306  
--target-version=8.0.18 --config-path=/etc/mysql/my.cnf
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

You can inspect the list of checks from the output of `util.checkForServerUpgrade()` function in the practice titled "Upgrading MySQL."

Using In-Place Upgrade Method

Use the in-place (physical) upgrade method when you upgrade within a series or from one major version to the next.

- Stop the MySQL server process.
- Use file copy to back up the current databases. (Optional, but recommended)
- Replace the `mysqld` binary with the new version.
- Start the MySQL server process using the new binary.
- Prior to MySQL 8.0.16, you must run `mysql_upgrade`. In MySQL 8.0.16 and higher, the server restart will perform all the upgrade tasks automatically.

This is the recommended method to upgrade from MySQL 5.7 to 8.0.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Using Logical Upgrade Method

Use the logical (backup/restore) upgrade method when you upgrade to one or more later major versions.

- Use `mysqldump` to back up the current databases.
- Install and initialize a new MySQL server.
- Start the new MySQL server.
- Restore the backed-up databases from the dump file.
 - May result in errors due to incompatibilities introduced by new, changed, deprecated, or removed features and capabilities
- Prior to MySQL 8.0.16, you must run `mysql_upgrade`. In MySQL 8.0.16 and higher, you must shut down the server and restart it with the `--upgrade=FORCE` option to perform the remaining upgrade tasks.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Using `mysqldump` to Perform a Logical Backup

The following command is the recommended way to perform a logical backup for use during the upgrade process:

```
mysqldump --all-databases --routines --events --add-drop-table  
          --lock-all-tables --flush-privileges=0 > data-for-upgrade.sql
```

Note: The process of backing up data by using `mysqldump` is covered in the lesson titled “Performing Backups.”

Other Methods

If you use replication, you might need to perform a rolling upgrade, where you upgrade hosts in turn without bringing down the replication topology at one time.

You might also reduce down time by running the application from a temporary host that you put in place for the duration of the upgrade.

`mysql_upgrade`

- Checks all tables in your databases for incompatibilities with current versions of the MySQL server
- Repairs any problems found in tables with possible incompatibilities
- Upgrades system tables to add any new privileges or capabilities that are available in the new version
- Marks all checked and repaired tables with the current MySQL version number
- Not required for MySQL server 8.0.16 or later. The server will perform all the upgrade processes during startup. A new option `--upgrade` is introduced in 8.0.16 to control the upgrade process.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Versions

During the upgrade process, you might have multiple versions of `mysql_upgrade` on the same host at the same time. Ensure that you execute the correct version of `mysql_upgrade` so that it assumes the correct target version of MySQL.

The `--upgrade` option has the following values:

- `AUTO` (default) : The server performs an automatic upgrade of anything it finds to be out of date.
- `NONE` : The server performs no automatic upgrade steps during the startup.
- `MINIMAL` : The server upgrades the data dictionary if necessary but does not upgrade anything else.
- `FORCE` : The server upgrades the data dictionary if necessary and forces an upgrade of everything else.

Summary



In this lesson, you should have learned how to:

- Install the MySQL server and client programs
- Identify the files and folders created during installation
- Perform the initial configuration of the MySQL server
- Start and stop MySQL
- Upgrade to MySQL 8.0



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Practices

- 2-1: Installing MySQL
- 2-2: Connecting to MySQL
- 2-3: Configuring the MySQL Service
- 2-4: Upgrading MySQL
- 2-5: Deploying MySQL with Docker



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Understanding MySQL Architecture

3



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Objectives



After completing this lesson, you should be able to:

- Explain how MySQL processes, stores, and transmits data
- Describe the Transactional Data Dictionary
- Configure InnoDB tablespaces
- Explain how MySQL uses memory
- Configure the InnoDB buffer pool
- List some of the available plugins



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- How MySQL Stores Data
- How MySQL Stores Metadata
- InnoDB Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin and Component Interface

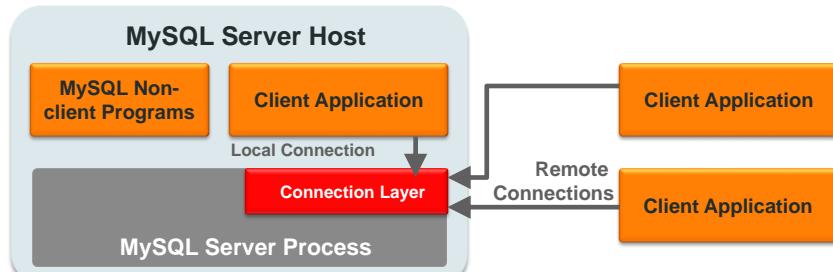


ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Architecture

- A MySQL installation has the following architectural components:
 - The MySQL server process
 - Client programs connecting locally or remotely
 - Other MySQL programs (that are not clients) installed locally
- Client programs connect to the MySQL server process to make data requests.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Client/Server Connectivity

MySQL client/server communication is not limited to environments where all computers run the same operating system.

- Client programs can connect to the server running on the same host or on a different host.
- Client/server communication can occur in environments where computers run different operating systems.
- Example:
 - MySQL server process running on Linux
 - Client programs running on Windows and connecting via TCP/IP



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL Server Process

- Is a program called `mysqld`
- Runs as a single process that is multithreaded
- Manages access to databases on disk and in memory
- Supports simultaneous client connections
- Supports multiple storage engines
- Supports both transactional and non-transactional tables
- Optimizes memory usage by using caches and buffers



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

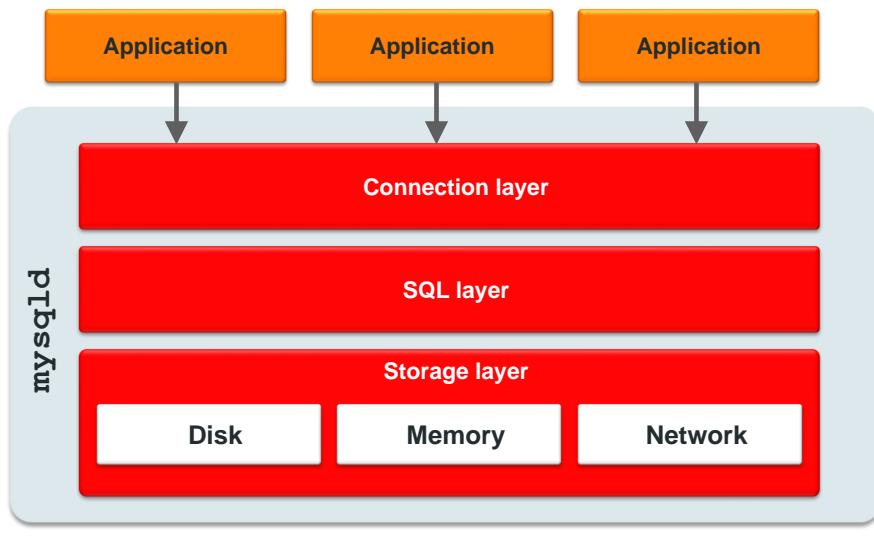
Terminology

- **Host:** The physical machine or virtual machine on which the server program runs, which includes the following:
 - Its hardware configuration
 - The operating system running on the machine
 - Its network addresses
- **MySQL server software:** A software program (`mysqld`) with a version number and a list of features
- **MySQL server instance:** A `mysqld` server process that manages a data directory containing one or more schemas
 - Multiple instances can run on a single host.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Server Process



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The mysqld (server program) process consists of three layers:

- The *Connection layer* handles connections. This layer is not unique to MySQL. All server software has such a layer (for example, web/mail/LDAP servers).
- The *SQL layer* processes SQL queries that are sent by connected applications.
- The *Storage layer* handles data storage in various formats and structures on various types of physical media.

Topics

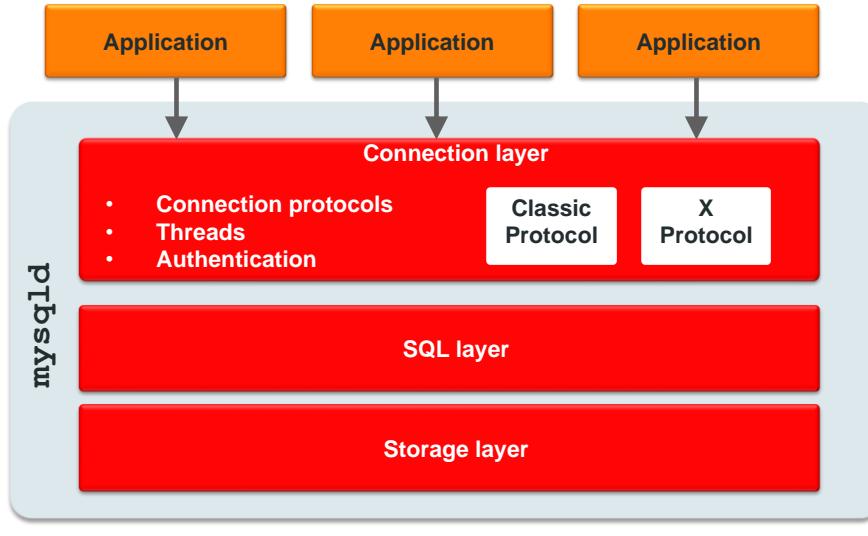
- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- How MySQL Stores Data
- How MySQL Stores Metadata
- InnoDB Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin and Component Interface



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Connection Layer



ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The connection layer maintains one thread per connection. This thread handles query execution. Before a connection can begin sending SQL queries, the connection is authenticated by the verification of username, password, and client host.

The connection layer accepts connections from applications over several connection protocols:

- TCP/IP
- UNIX sockets
- Shared memory
- Named pipes

MySQL 8.0 supports two types of connections:

- MySQL Classic Protocol
- MySQL X Protocol

Connection Protocols

- Protocols are implemented in the client libraries and drivers.
- The speed of a connection protocol varies with the local settings.
- In addition to the legacy MySQL Classic Protocol, the MySQL X Protocol is introduced in MySQL 5.7.12 and enabled by default in MySQL 8.0.

Protocol	Connection	Supports	Operating Systems
TCP/IP	Local, remote	Classic and X Protocol	All
Socket file	Local	Classic and X Protocol	UNIX-derived operating systems including Linux, BSD, Max OS X
Shared memory	Local	Classic Protocol	Windows
Named pipes	Local	Classic Protocol	Windows



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL uses TCP to transmit messages from clients to the server over the network. The other protocols listed in the slide support only local connections where the client and server are running on the same machine.

The MySQL X Protocol can be configured by using variables and options prefixed with `mysqlx`. Some examples of `mysqlx` variables:

- `mysqlx`
- `mysqlx_bind_address`
- `mysqlx_max_connections`
- `mysqlx_port`
- `mysqlx_socket`

Local and Remote Connection Protocol: TCP/IP

- TCP/IP (Transmission Control Protocol/Internet Protocol):
 - Is the suite of connection protocols used to connect hosts on the Internet
 - Uses IP addresses or DNS host names to identify hosts
 - Uses TCP port numbers to identify specific services on each host
 - MySQL default TCP port number:
 - **3306** for MySQL Classic protocol (server `port` option)
 - **33060** for MySQL X Protocol (server `mysqlx_port` option)
 - **33062** for administrative connection using MySQL Classic protocol (server `admin_port` option)
 - Enables connections between different hosts
- Example using a host name and the default port (no option specified):

```
mysql --host=mysqlhost1 --user=root --password
```

- Example using an IP address and an alternative port (use the `-P` or `--port` option):

```
mysql -h 192.168.1.8 -P 3309 -uroot -p
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

As of MySQL 8.0.14, MySQL server permits a TCP/IP port to be configured specifically for administrative connections. This provides an alternative to the single administrative connection that is permitted on the network interfaces used for ordinary connections even when `max_connections` connections are already established. The interface is available only if the `admin_address` system variable is set at startup to indicate the IP address for the administrative interface. If no `admin_address` value is specified, the server maintains no administrative interface. Connections are permitted only by users who have the `SERVICE_CONNECTION_ADMIN` privilege. There is no limit on the number of administrative connection.

MySQL server uses DNS (Domain Naming System) to resolve the names of client hosts that connect using the TCP/IP protocol, storing them in a host cache. For large networks that exhibit performance problems during name resolution, disable DNS with the `--skip-name-resolve` option or increase the value of the `--host-cache-size` option.

Local Connection Protocol in Linux: Socket

- A form of inter-process communication
 - Used to form one end of a bidirectional communication link between two processes on the same machine
- Requires that the server creates a socket file on the local system with the `socket` and `mysqlx_socket` options for MySQL classic and X Protocol, respectively
 - The client specifies the socket file with the option `--socket` or `-S` when it connects.
 - This is the best connection type for Linux.
- Example using the `/var/lib/mysql/mysql.sock` socket file:

```
mysql -s /var/lib/mysql/mysql.sock -uroot -p
```

- Example using the default socket file `/tmp/mysql.sock`:

```
mysql -uroot -p
```

- If no host is specified, `mysql` assumes `-h localhost`.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL and localhost

- MySQL assumes the TCP protocol when you specify `--host=hostname` except when the host name is `localhost`.
 - If the host name is `localhost`, MySQL assumes you are connecting using a UNIX socket.
- To connect to the `localhost` IP address `127.0.0.1` or `::1`:
 - Specify TCP explicitly:

```
mysql -h localhost --protocol=tcp -uroot -p
```

– Specify the `localhost` IP address explicitly:

```
mysql -h 127.0.0.1 -uroot -p
```

```
mysql -h ::1 -uroot -p
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Operating System Support for IPv6

Although MySQL supports IPv6 connections, the IPv6 protocol requires operating system support. You can test operating system support in Linux by using the `ping6` command:

```
ping6 ::1
```

Local Connection Protocols in Windows: Shared Memory and Named Pipes

- Shared memory:
 - The server creates a named shared memory block. Client processes running on the same host uses the shared memory to communicate with the server.
 - Shared memory is disabled by default.
 - Example of using the default shared memory base name MYSQL:

```
mysql --protocol=memory -uroot -p
```

- Named pipes:
 - In Windows, named pipes work much like UNIX sockets:
 - The server creates the named pipe, and the client writes to it and reads from it.
 - Named pipes support read/write operations, along with an explicit passive mode for server applications.
 - Example:

```
mysql --protocol=pipe -uroot -p
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Disabled by Default

Both shared memory and named pipe connection protocols are disabled by default in MySQL servers.

- To enable shared-memory connections, start the server with the `--shared-memory` option.
- To enable named-pipe connections, you must start the server with the `--enable-named-pipe` option.

X Protocol Support

Both the shared memory and named pipes in Windows do not support the MySQL X Protocol. They work with the MySQL Classic Protocol only.

SSL by Default

For TCP/IP connections, the connection layer uses secure, encrypted connections when they are available.

- MySQL package installers create SSL keys if OpenSSL is installed on the server host.
 - The installer calls the `mysql_ssl_rsa_setup` utility to create the keys.
 - If you install from a binary archive, call this utility manually.
- MySQL clients use SSL if keys are available.
 - Keys are in the data directory.
 - Copy client keys to remote clients to enable encrypted remote connections.
- If SSL is not available, connections are unencrypted.
 - You can configure the server and clients to use SSL mandatorily.
- SSL is not supported on socket, named pipe, and shared memory connections.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

When you create a user on the server, you can configure the user to use SSL connections mandatorily. The server disallows connections from such a user if the connection is not encrypted.

Note: SSL keys are covered in more detail in the lesson titled “Securing MySQL.”

Connection Threads

- The server creates a connection thread for each active client connection.
 - All statements executed by that client are executed by a single server thread.
 - The server destroys the thread when the client disconnects.
- The Thread Pool plugin manages connections and server threads separately:
 - Manages client connections with thread groups
 - Each thread group allows only one short-running statement at any point in time.
 - Reduces the number of server threads
 - A thread group can create additional server threads for long-running statements.
 - Distinguishes between high-priority and low-priority statements based on their transaction membership
 - Statements within a running transaction are high priority.
 - Long-running transactions are improved.
- X DevAPI supports connection pooling, where the client can maintain a pool of connections for reuse.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Performance Overhead

When the server creates and destroys connection threads, it must allocate and deallocate memory structures used by those client connections. There is a performance overhead associated with this activity when clients connect and disconnect frequently.

The Thread Pool plugin is available in commercial editions of MySQL. It is designed to reduce the performance overhead of creating and destroying connection threads by:

- Separating the client connection from the connection thread
- Limiting the number of statements that execute concurrently, especially short-running statements

Connection pooling can reduce the overhead for applications that open many connections to a MySQL server. See <https://dev.mysql.com/doc/x-devapi-userguide/en/connecting-connection-pool.html> for the options to enable connection pooling.

Quiz



Which of the following parameters to the `--protocol` option works on all operating systems, both locally and remotely?

- a. PIPE
- b. MEMORY
- c. SOCKET
- d. TCP



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Answer: d

Topics

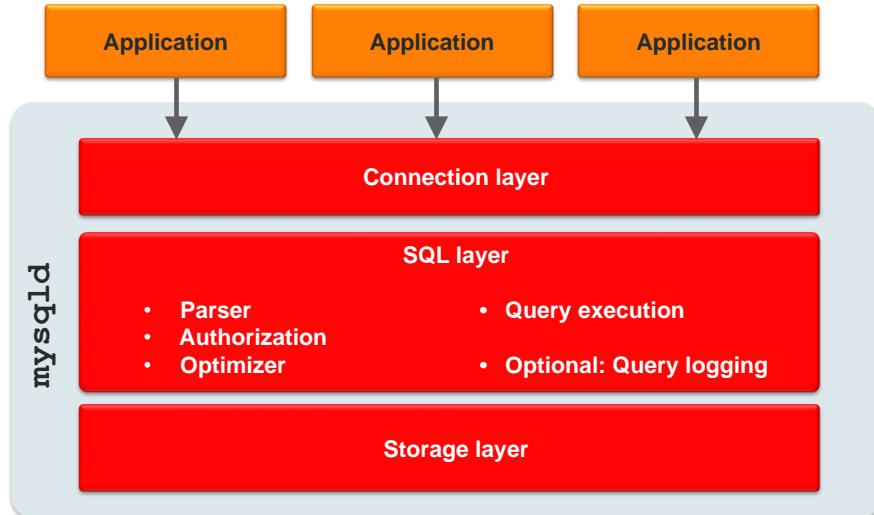
- Architectural Overview
- How MySQL Transmits Data
- **How MySQL Processes Requests**
- How MySQL Stores Data
- How MySQL Stores Metadata
- InnoDB Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin and Component Interface



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

SQL Layer



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

SQL Layer Components

After a connection is established, MySQL processes each query in the SQL layer, which comprises the following components:

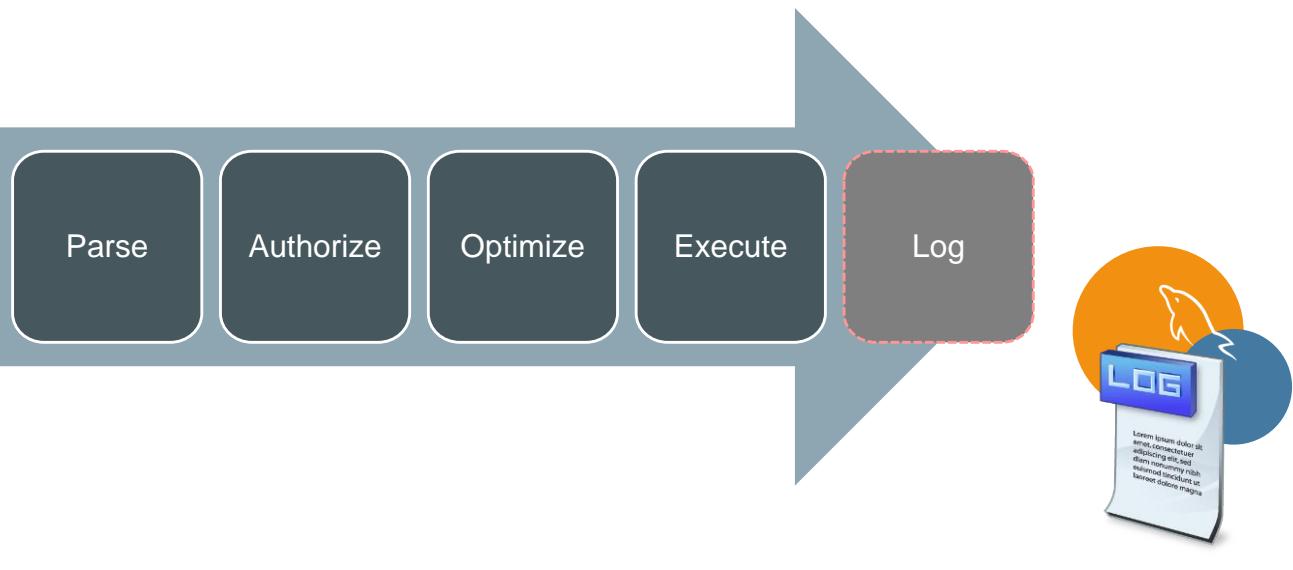
- **Parser:** Validates the query's syntax and semantics and converts it to a standard form
- **Authorization:** Verifies that the connected user is allowed to run the query and has enough permissions on the objects the query refers to
- **Optimizer:** Creates an optimal execution plan for each query. This involves deciding which indexes to use and in which order to process the tables
- **Query execution:** Fulfills the execution plan for each query
- **Query logging:** Logs queries that the server receives or executes



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

SQL Statement Processing



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The SQL layer parses, authorizes, optimizes, and executes queries in that order. If you have enabled any of the logs, it also logs the query.

Topics

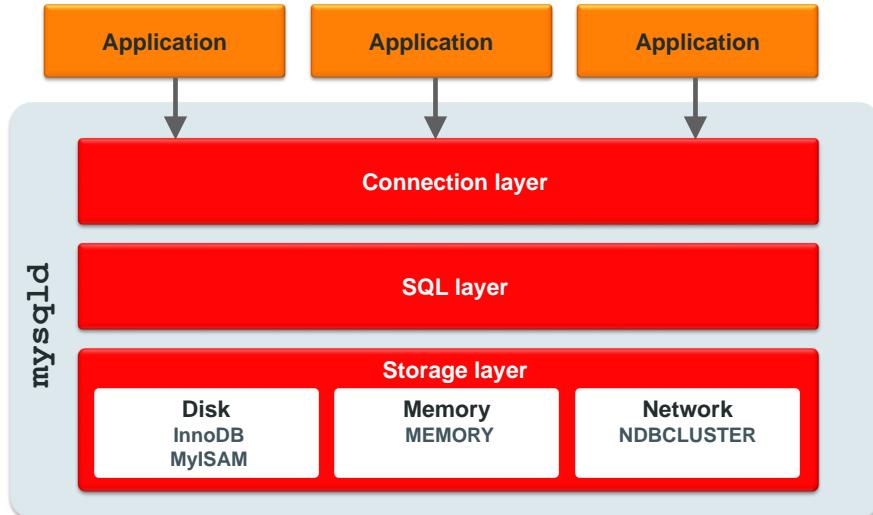
- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- **How MySQL Stores Data**
- How MySQL Stores Metadata
- InnoDB Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin and Component Interface



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Storage Layer



ORACLE®

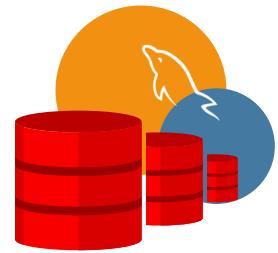
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Storage Engines

InnoDB, MyISAM, MEMORY, and NDBCLUSTER are examples of storage engines.

Storage Engines Provided with MySQL

- InnoDB:
 - This is the default, built-in storage engine.
 - Use this engine except in specific, rare circumstances.
- Other engines included with MySQL:
 - MyISAM (often used in legacy systems)
 - MEMORY
 - ARCHIVE
 - BLACKHOLE
 - MERGE
 - CSV
 - FEDERATED (disabled by default)
 - NDBCLUSTER (available in MySQL Cluster distributions)
- Third-party storage engines are also available.



ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

InnoDB and NDBCLUSTER are the only two MySQL storage engines that are transactional and that support foreign keys.

The EXAMPLE storage engine is included in the MySQL source code and is intended to be useful to storage engine developers.

Third-party engines have different sources and features and are not supported by MySQL. For further information, documentation, installation guides, bug reporting, or any help or assistance with these engines, contact the developer of the engine directly.

Storage Engines: Function

- Storage engines are server components that act as handlers for different table types.
- MySQL delegates the task of handling data rows to these storage engines, which:
 - Store the data on disk, memory, or other components on the network
 - Provide indexes and other row optimizations
- When you create a table, you specify which of the available storage engines manages its data.
- The table's storage engine does not usually affect the operation of the SQL layer.
 - In general, the SQL layer parses all valid SQL and the storage layer handles row operations.
 - Exceptions are covered in the next slide.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

A storage engine is a low-level component that takes care of storing and retrieving data and can be accessed through an internal MySQL API or, in some situations, directly by an application. An application might use tables that use different storage engines.

SQL and Storage Layer Interactions

- SQL statements are storage-engine independent, apart from the following:
 - `CREATE TABLE` has an `ENGINE` option that specifies which storage engine to use on a per-table basis.
 - `ALTER TABLE` has an `ENGINE` option that enables the conversion of a table to use a different storage engine.
- Some features are available in only some storage engines. For example:
 - Only InnoDB and NDB support:
 - Foreign keys
 - Transaction control operations, such as `COMMIT` and `ROLLBACK`
 - Table partitioning (starting from MySQL 8.0)
 - Only InnoDB and MyISAM support full-text indexes and R-tree spatial indexes.
 - Only MEMORY and NDB support hash indexes.
 - Only InnoDB supports Transparent Data Encryption (TDE).



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Parsing Storage Engine-Specific Syntax

The SQL layer must process statements before passing row operations to the storage layer. This means that it must parse statements that are not supported by some storage engines, and in some cases, it returns a warning if the storage layer does not support the specified operation.

For example, MyISAM does not support foreign keys. So for MyISAM tables, the SQL layer parses `FOREIGN KEY` definitions in tables, but the storage layer ignores them. For transactional operations, such as `ROLLBACK`, MySQL generates a warning when the storage engine is not transactional. The server will return an error if you try to execute full text search queries on a non-supported storage engine.

Features Dependent on Storage Engine

- Storage medium
 - Disk
 - Memory
 - Networked data nodes
 - Null (BLACKHOLE)
- Transactional capabilities
 - Multistatement transactions with commit and rollback
 - Isolation levels
- Locking
 - Locking granularity beyond table level
 - Multiversion Concurrency Control (MVCC)
- Table Partitioning
 - Becomes engine specific in MySQL 8.0



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Features Dependent on Storage Engine

- Backup and recovery
 - Complex storage engines, such as InnoDB and NDB, maintain consistency internally to improve performance.
 - Files do not always contain a consistent snapshot of the running database.
 - Filesystem (raw) backups are possible with simpler engines.
 - Example: MyISAM has a few consistency features, so the files contain a consistent view of the database.
- Optimization
 - Some storage engines use indexes, internal caches, buffers, and memory to optimize performance.
- Referential integrity with foreign keys
- Full-text search
- Spatial data



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

InnoDB Storage Engine

InnoDB, the default storage engine for MySQL, provides high reliability and high performance, as well as the following primary advantages:

- Transaction-safe (ACID compliant)
- MVCC (Multiversioning Concurrency Control)
 - InnoDB row-level locking
 - Oracle-style consistent non-locking reads
- Table data arranged to optimize primary key-based queries
- Support for foreign key referential integrity constraints
- Maximum performance on large data volumes
- Fast auto-recovery after a crash
- Buffer pool for caching data and indexes in memory
- Support for table partitioning



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The following are some additional advantages of using InnoDB:

- **Transaction-safe:** ACID compliance is achieved with transaction commit, rollback, and crash-recovery capabilities to protect user data.
- **Foreign key support:** Includes cascaded deletes and updates
- **Spatial data and indexing:** Efficient storage and retrieval of geographic information (GIS) data
- **Full-text indexing:** Enables efficient searching for words or phrases within text columns
- **Buffer pool warmup:** Saves a percentage of the most recently used pages in the buffer pool at server shutdown and restores these pages at server startup
- **Instant ADD COLUMN:** From MySQL 8.0.12 onward, add a column online by modifying the metadata only without the need to rebuild the table data. See <https://dev.mysql.com/doc/refman/8.0/en/innodb-online-ddl-operations.html#online-ddl-column-syntax-notes> for the limitations of this feature.

MyISAM Storage Engine

- Is used in many legacy systems
 - Was the default storage engine before MySQL 5.5
- Is fast and simple, but subject to table corruption if server crashes
 - Use `REPAIR TABLE` to recover corrupted MyISAM tables.
- Supports `FULLTEXT` indexes
- Supports spatial data types and indexes
- Supports table-level locking
- Supports raw table-level backup and recovery because of the simple file format
- No transactional support
- No support for table partitioning in MySQL 8.0 as compared to MySQL 5.7.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MyISAM was the default MySQL storage engine before version 5.5.5, when it was superseded by InnoDB. MyISAM supports a limited set of features. If you work with a system that uses MyISAM tables, consider changing them to InnoDB.

MEMORY Storage Engine

- Stores row data and indexes in memory
 - Data does not survive server restarts.
- Stores rows with a fixed-length format
- Support binary trees and hash indexes
- Limits table size with the `--max-heap-table-size` option
 - The option is named for the older storage engine name HEAP.
- Supports table-level locking
- Cannot store `TEXT` or `BLOB` columns



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MEMORY engine performance is constrained by contention resulting from single-thread execution and table-lock overhead when processing updates. This limits scalability when the load increases, particularly for statement mixes that include writes. Also, MEMORY engine does not preserve table contents across server restarts.

If you work with a system that uses MEMORY tables, consider changing them to InnoDB, which has a buffer pool that automatically stores frequently accessed data in memory. If you have InnoDB tables that are small and frequently accessed enough to use as MEMORY tables, then the InnoDB buffer pool might contain all rows in those tables. This provides all of the benefits of using memory without any of the drawbacks.

ARCHIVE Storage Engine

The ARCHIVE storage engine is used for storing large volumes of data in a compressed format, allowing for a very small footprint. It has these primary characteristics:

- Does not support indexes
- Supports INSERT and SELECT, but not DELETE, REPLACE, or UPDATE
- Supports ORDER BY operations and BLOB columns
- Accepts all data types except spatial data types
- Uses row-level locking
- Supports AUTO_INCREMENT columns
- Disabled by default, need to be enabled to use



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

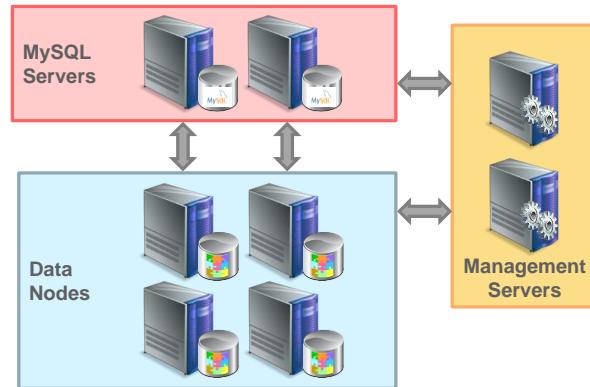
The ARCHIVE engine supports a limited set of features. If you work with a system that uses ARCHIVE tables, consider changing them to InnoDB compressed tables.

InnoDB and ARCHIVE compression both use the zlib library, although due to the overhead of the InnoDB file format, ARCHIVE file size is typically smaller than an equivalent compressed InnoDB file containing the same data.

NDBCluster Storage Engine

The NDBCluster storage engine (also known as NDB) enables running several computers with MySQL servers, data nodes, and management servers in a cluster. It offers these primary characteristics:

- Only available with MySQL Cluster
- Supports a “cluster” (group of nodes working together to store/retrieve data)
- Runs outside of the MySQL server as one or more processes
- MySQL server provides SQL interface to the cluster processes.
- Supports transactions and foreign keys
- Provides high availability, scalability, and high performance



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

For more information about NDBCluster storage engine features, see
<http://dev.mysql.com/doc/refman/en/mysql-cluster.html>

BLACKHOLE Storage Engine

- Acts as a null storage engine
 - It accepts data but does not store it.
 - Retrievals always return an empty result.
- Supports all kinds of indexes
- Is useful in some specific cases:
 - Replication: Relay slaves that log and forward replicated logs but do not store the data.
 - Committed transactions are written to the binary log, but rolled-back transactions are not.
 - Verifying backups and dump file syntax
 - Finding performance bottlenecks not related to the storage engine
 - Example: Measuring the overhead from binary logging



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Note

For more information about binary logging and replication, see the lessons titled “Configuring a Replication Topology” and “Administering a Replication Topology.”

Storage Engines Feature Summary

Feature	InnoDB	MyISAM	Memory	Archive	NDBCluster
B-tree indexes	Yes	Yes	Yes	No	No
Backup/point-in-time recovery [1]	Yes	Yes	Yes	Yes	Yes
Cluster database support	No	No	No	No	Yes
Clustered indexes	Yes	No	No	No	No
Compressed data	Yes	Yes [2]	No	Yes	No
Data caches	Yes	No	N/A	No	Yes
Transparent Data Encryption	Yes	No	No	No	No
Foreign key support	Yes	No	No	No	Yes [3]
Full-text search indexes	Yes [4]	Yes	No	No	No
Geospatial data type support	Yes	Yes	No	Yes	Yes
Geospatial indexing support (R-tree)	Yes [5]	Yes	No	No	No



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Notes

- [1] Implemented in the server, rather than in the storage engine.
- [2] Compressed MyISAM tables are supported only when using the compressed row format. Tables using the compressed row format with MyISAM are read only.
- [3] Support for foreign keys is available in MySQL Cluster NDB 7.3 and later.
- [4] InnoDB support for FULLTEXT indexes is available in MySQL 5.6 and later.
- [5] InnoDB support for geospatial indexing is available in MySQL 5.7 and later.

Storage Engines Feature Summary

Feature	InnoDB	MyISAM	Memory	Archive	NDBCluster
Table Partitioning [6]	Yes	No	No	No	Yes
Hash indexes	No [7]	No	Yes	No	Yes
Index caches	Yes	Yes	No	No	Yes
Locking granularity	Row	Table	Table	Row	Row
Multiversion Concurrency Control (MVCC)	Yes	No	No	No	No
Replication support [1]	Yes	Yes	Limited [8]	Yes	Yes
Storage limits	64TB per tablespace	256TB per table	RAM	None	384EB
T-tree indexes	No	No	No	No	Yes
Transactions	Yes	No	No	No	Yes
Update statistics for data dictionary	Yes	Yes	Yes	Yes	Yes



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Notes

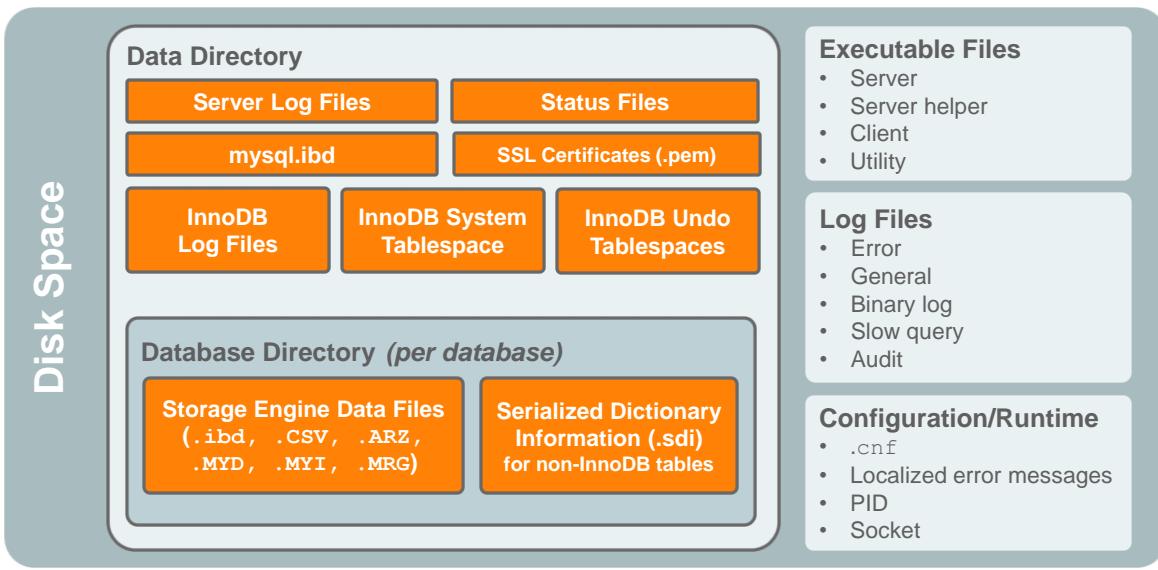
[1] Implemented in the server, rather than in the storage engine.

[6] MySQL 8.0 implements table partitioning in the storage engine, rather than in the server.

[7] InnoDB utilizes hash indexes internally for its Adaptive Hash Index feature.

[8] To synchronize master and slave MEMORY tables, when a MEMORY table is used on a master for the first time since it was started, a `DELETE` statement is written to the master's binary log, to empty the table on the slaves also.

How MySQL Uses Disk Space



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Note

Serialized Dictionary Information is covered in the section "How MySQL Stores Metadata" later in this lesson.

Data Directory

- The primary use of disk space is the data directory.
- The location of the data directory is configurable.
 - The default location in Linux is `/var/lib/mysql`.
- Server log files and status files contain information about statements that the server processes. Logs are used for troubleshooting, monitoring, replication, and recovery.
 - InnoDB redo log files and undo tablespaces for all databases reside at the data directory level.
- InnoDB system tablespace contains the doublewrite buffer and the change buffer.
- Each database (including the `mysql` system database) has a single directory under the data directory that contains storage-engine specific data files for the database.
 - For example: `tablename.ibd` for InnoDB, which contains the table data and metadata
 - Serialized Dictionary Information (`.sdi`) metadata files for other storage engines



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

mysql Database

MySQL stores the `mysql` system database on disk just like any other database. The `mysql` database contains information such as users, privileges, plugins, help topics, and time-zone data.

All the InnoDB tables in the `mysql` system database are stored in the `mysql` general tablespace (`mysql.ibd`) at the data directory level. Non InnoDB tables are stored in the `mysql` database directory.

Serialized Dictionary Information

InnoDB tables store metadata information as serialized dictionary information (SDI) in the `.ibd` file together with the data. Other storage engines store this in a separate `.sdi` file in JSON (JavaScript Object Notation) format.

Topics

- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- How MySQL Stores Data
- **How MySQL Stores Metadata**
- InnoDB Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin and Component Interface



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

What Is a Data Dictionary?

Metadata is information about the data stored in an RDBMS, such as:

- Table and column definitions
- Index and constraint definitions
- User and privileges information

ID	Name	Department	Salary
1	Sarah Finch	Sales	100000



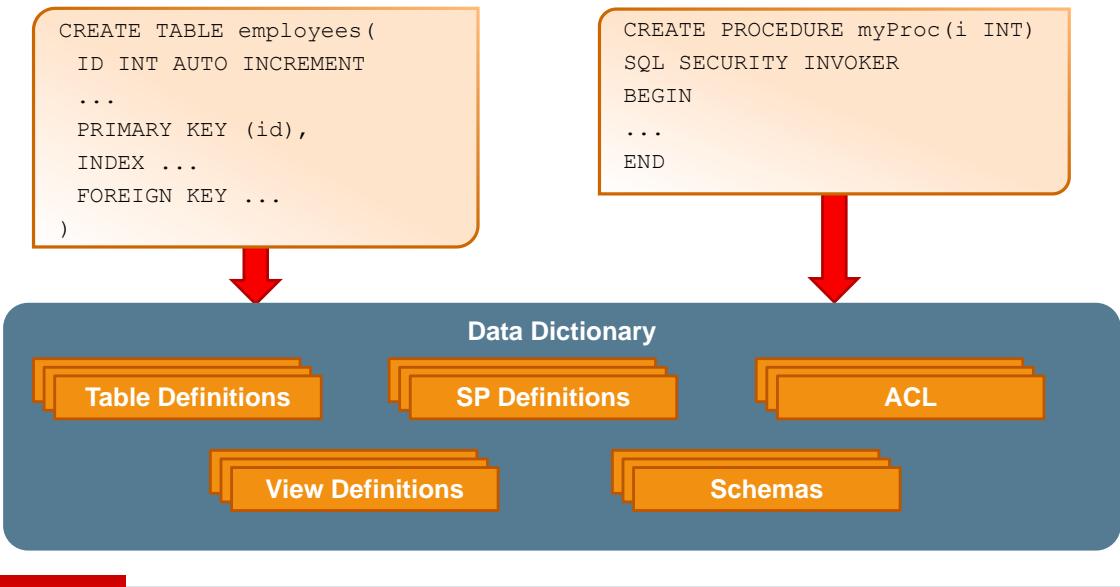
- The data dictionary is a centralized repository of metadata for all the data in an RDBMS.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The data dictionary stores metadata in an RDBMS. Metadata is information about the data that the database stores and includes things like column definitions, foreign key constraints, index definitions, and so on. The data dictionary is a collection of *all* this data for the RDBMS.

Types of Metadata

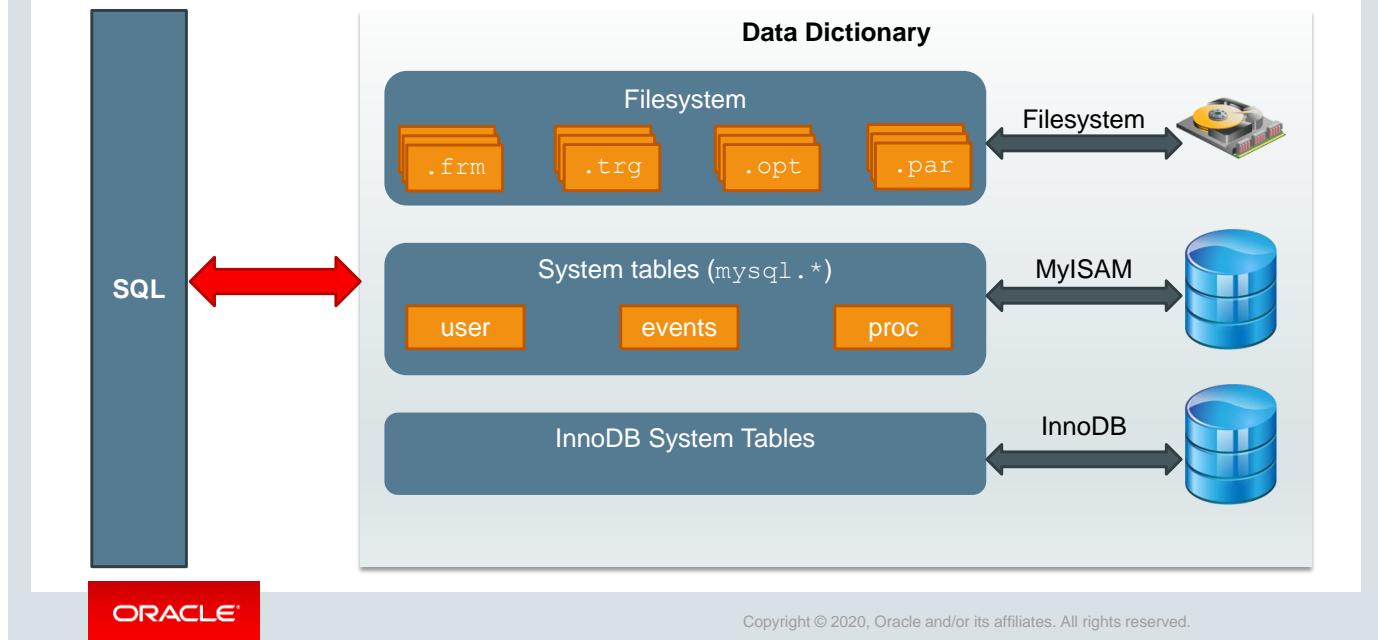


ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

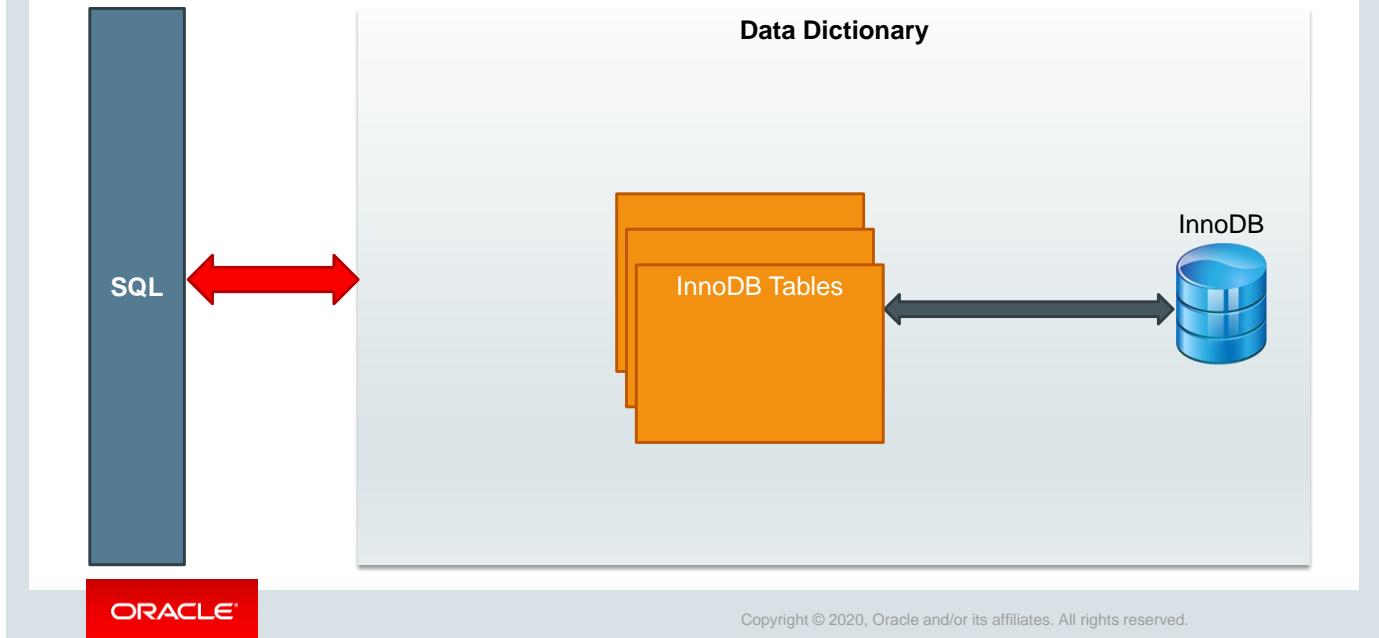
When you create a database object, its metadata is stored in the data dictionary. For example, if you create a table, then information about its columns, indexes, and so on is stored. If you create a stored procedure, then the data dictionary stores metadata such as invokers and privileges in the Access Control List (shown as ACL in the diagram).

Data Dictionary in Earlier Versions of MySQL



Before MySQL 8.0, the data dictionary stored metadata in different locations. For example, .frm files for table definitions, .opt files for databases, .par files for partitioning, .trg and .trn files for triggers, and so on were all stored in the host filesystem. Events, stored procedures, the user tables, and access control information were stored in non-transactional MyISAM tables, and InnoDB also maintained its own metadata. This led to inconsistencies, issues with replication, and data that was not "crash safe."

Transactional Data Dictionary in MySQL 8



In MySQL 8.0, all metadata is stored in InnoDB tables. There are no .frm, .opt, or other metadata files in the filesystem and no reliance on MyISAM, which is a non-transactional storage engine.

Transactional Data Dictionary: Features

- Single metadata repository for all MySQL server subsystems
 - All storage engines have their own user tables, but all their metadata are stored in the same data dictionary tables.
 - The tables reside in the InnoDB Data Dictionary tablespace.
- Based on standard SQL definitions
 - Easier to extend: Common Data Dictionary API
 - Automatic upgrades by using the installer
- Uses the transactional storage engine, InnoDB
 - Atomic DDL
 - Crash-safe
- Improves INFORMATION_SCHEMA
 - Better performance using standard optimization techniques
 - Easier to maintain



ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

With the introduction of the data dictionary, some of the INFORMATION_SCHEMA tables are implemented as views on data dictionary tables. You can query the INFORMATION_SCHEMA tables to view the database metadata, information about the MySQL server such as the name of a database or table, the data type of a column, or access privileges.

Serialization of the Data Dictionary

Every time there is a change to the metadata, MySQL creates a copy of it:

- The metadata is serialized in JSON format.
- Known as SDI (Serialized Dictionary Information)

MySQL stores the copy:

- InnoDB: In the InnoDB user tablespaces, along with the data
- MyISAM: As an `.sdi` file in the database directory



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL 8.0 provides crash safety by serializing the metadata whenever it changes. The output of this is in JSON (JavaScript Object Notation) format and is called Serialized Dictionary Information (SDI).

For InnoDB tables, this SDI is stored with the data in the InnoDB user tablespace. For MyISAM and other storage engines, it is written as an `.sdi` file in the data directory.

Note that this SDI is just a backup of the metadata. It is not the metadata itself. The data dictionary lives entirely within the InnoDB Data Dictionary tablespace.

Dictionary Object Cache

- Is a shared global cache that stores previously accessed data dictionary objects in memory to enable object reuse and minimize disk I/O
- Uses an LRU-based eviction strategy to evict least recently used objects from memory
- Comprises cache partitions that store different object types:

Cache partition	Variable to configure the limit	Object limit
tablespace definition	tablespace_definition_cache	Default is 256
schema definition	schema_definition_cache	Default is 256
table definition	max_connections	Default is 151
stored program definition	stored_program_definition_cache	Default is 256
character set definition		Hardcoded to 256
collation definition cache partition		Hardcoded to 256



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- How MySQL Stores Data
- How MySQL Stores Metadata
- **InnoDB Tablespaces**
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin and Component Interface



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

InnoDB Tablespaces

InnoDB tablespaces are data files that can store one or more InnoDB tables and associated indexes. InnoDB uses the following types of tablespaces:

- Data tablespaces
 - System tablespace
 - File-per-table tablespaces
 - General tablespaces
- Undo tablespaces
- Temporary table tablespaces



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

InnoDB System Tablespace

- InnoDB stores the change buffer and doublewrite buffer in the system tablespace. It may also contain table and index data.
- The `innodb_data_file_path` option configures the size and physical location of the InnoDB system tablespace files on disk.
 - Default value: `ibdata1:12M:autoextend`
 - One file called `ibdata1`, 12 MB in size, autoextending
 - Example value: `ibdata1:20M;/ext/ibdata2:10M:autoextend`
 - Two files:
 - `ibdata1` in the data directory, fixed at 20 MB in size
 - `ibdata2` in the `/ext` directory, 10 MB in size, auto-extending
 - If you have a set of files in the system tablespace, only the last file in the set can be auto-extending. The file can grow but does not shrink.
- The `innodb_file_per_table` option specifies whether MySQL stores new table data and indexes in the system tablespace or in a separate `.ibd` file.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

InnoDB system tablespace consists of:

- Change buffer: Changes to secondary index pages
- Doublewrite buffer: Ensures crash-safe writes
- Possibly table and index data of user-defined tables

It is strongly recommended to store the table data in file-per-table tablespace or general tablespace instead of the system tablespace for easier maintenance.

File-per-Table Tablespaces

- Are enabled by default
- Contain data and indexes from a single table, including its metadata
 - .ibd files named for the table in the database directory
- Example:



```
CREATE TABLE fpt_table(a INT PRIMARY KEY, b CHAR(4));
```

- Creates the fpt_table.ibd file in the current database's directory

- Example using an explicit TABLESPACE clause:

```
CREATE TABLE fpt_table(a INT PRIMARY KEY, b CHAR(4))
  TABLESPACE=innodb_file_per_table;
```

- Creates a file-per-table tablespace even if innodb_file_per_table is OFF

ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

General Tablespaces

- Contain data and indexes from multiple tables
 - .ibd files named when you create the tablespace
 - Do not belong to any particular database.
- Example:



```
CREATE TABLESPACE myts ADD DATAFILE 'myts_data1.ibd';
```

- Creates the myts_data1.ibd file in the data directory

- Place new tables in a general tablespace by specifying a TABLESPACE clause:

```
CREATE TABLE gen_table(a INT PRIMARY KEY, b CHAR(4)) TABLESPACE=myts;
```

- Move a table to a general tablespace with ALTER TABLE:

```
ALTER TABLE fpt_table TABLESPACE=myts;
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

InnoDB general tablespaces support only one data file per tablespace. You cannot add a second data file to an existing general tablespace.

Choosing Between File-Per-Table and General Tablespaces

- Per-table tablespaces provide the following benefits:
 - Table compression: You cannot mix compressed and uncompressed tables within the same general or system tablespace.
 - Space reclamation (with TRUNCATE): InnoDB drops and re-creates truncated file-per-table tablespaces, releasing free space back to the filesystem.
- General tablespaces provide the following benefits:
 - Less filesystem overhead for statements that remove large amounts of data
 - Such as DROP TABLE or TRUNCATE TABLE
 - Consume less memory to store tablespace metadata
- You can mix tablespace types within a database
 - Some tables using file-per-table
 - Some tables using general tablespaces
 - Multiple general tablespaces



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Locating Tablespaces Outside the Data Directory

- Place tablespaces outside the data directory to:
 - Physically separate datasets from other data
 - Place some data on faster devices than general data
- Use the DATA DIRECTORY clause when you create a table:

```
CREATE TABLE ext_table(a INT PRIMARY KEY, b CHAR(4))
    DATA DIRECTORY='/datadir2';
```

- Use a relative or absolute path when you specify the data file for a general tablespace.
 - Relative paths use the data directory as their root.
 - Absolute paths can refer to any valid filesystem target.

```
CREATE TABLESPACE ext_ts ADD DATAFILE '/datadir2/ext_ts_data1.ibd';
```

- The directory must be specified in the read only variable innodb_directories.
 - The directories specified in innodb_directories are also used for tablespace discovery during crash recovery.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

innodb_directories may be specified as an option in a startup command or in a MySQL option file. A restart is required to change the value. Directories defined by innodb_data_home_dir, innodb_undo_directory, and datadir are automatically appended to the innodb_directories argument value, regardless of whether the innodb_directories option is specified explicitly.

Temporary Tablespaces

InnoDB has two types of temporary tablespaces:

- **Session Temporary Tablespaces:** Store user-created temporary tables and internal temporary tables created by the optimizer:
 - Allocated to a session from a pool of temporary tablespaces on the first request to create an on-disk temporary table
 - A pool of ten temporary tablespaces is created when the server is started. The size of the pool never shrinks, and tablespaces are added to the pool automatically as necessary.
- **Global Temporary Tablespace:** Stores rollback segments (undo logs) for changes made to user-created temporary tables:
 - The relative path, name, size, and attributes for temporary tablespace files depend on the value of `innodb_temp_data_file_path`.
 - If no value set, all temporary files are created in a 12 MB auto-extending data file called `ibtmp1` in the `innodb_data_home_dir` directory.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Session Temporary Tablespaces

A maximum of two tablespaces is allocated to a session, one for user-created temporary tables and the other for internal temporary tables created by the optimizer. When a session disconnects, its temporary tablespaces are truncated and released back to the pool.

Each tablespace is stored as a file with extension `.ibt` in the directory specified by the `innodb_temp_tablespaces_dir` variable. The default location is the `#innodb_temp` directory in the data directory.

- The `INFORMATION_SCHEMA.INNODB_SESSION_TEMP_TABLESPACES` table provides metadata about session temporary tablespaces.
- The `INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO` table provides metadata about user-created temporary tables that are active in an InnoDB instance.

Topics

- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- How MySQL Stores Data
- How MySQL Stores Metadata
- InnoDB Tablespaces
- **Redo and Undo Logs**
- How MySQL Uses Memory
- Plugin and Component Interface



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Redo Logs

- Store InnoDB change operations before they are made to the data files
 - Enables InnoDB to optimize data writes so that they do not need to occur synchronously
- Are used during crash recovery
 - InnoDB replays operations in the redo log files to ensure transactional consistency across all tables, even for operations that did not write to the data files before the crash.
- The number of files and file size are controlled by the `innodb_log_files_in_group` and `innodb_log_file_size` options, respectively.
- Are located in the data directory by default
 - Typically, the files are named `ib_logfile0` and `ib_logfile1`
 - If the files have wrong size or corrupted, InnoDB fails to start.
 - If the files do not exist, they are created during the server startup.
 - Controlled by the `innodb_log_group_home_dir` option

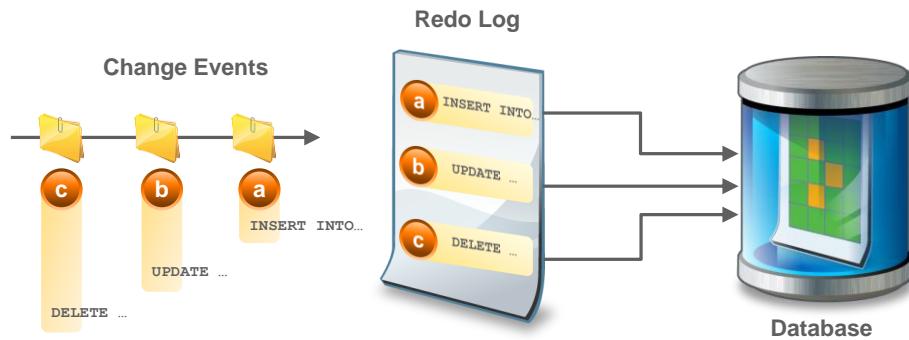


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The combined size of log files (`innodb_log_file_size * innodb_log_files_in_group`) can be up to 512 GB.

The default and recommended number of files is 2. Having large redo log files may increase the memory usage for I/O buffer at the OS level. It can be beneficial to have more files that are smaller in size to reduce the memory usage.

Redo Logs



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

As change events arrive, the redo log records the events before writing the changes to the database. For example, event (a) might be an `INSERT INTO` operation and event (b) an `UPDATE`. InnoDB writes the effects of the change events to the database pages only after it writes to the redo log.

Undo Logs

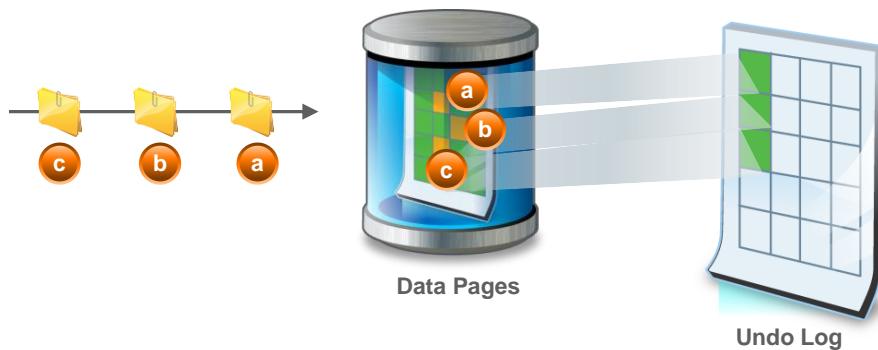
- Store copies of unmodified data that becomes modified by transactions so that InnoDB can access an earlier version of the data
- Are also called *rollback segments*
- Are stored by default in the *undo tablespace*
- Are used for MVCC and rollback
 - InnoDB retrieves the unmodified data from the undo log:
 - If you roll back a transaction
 - If another transaction needs to see earlier data as part of a consistent read
- Internally split into:
 - Insert undo buffer
 - Update undo buffer



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The redo and undo logs are distinct from other MySQL logs, including the general query log, slow query log, binary log, and audit log. You cannot inspect the redo and undo logs by using standard tools.

Undo Logs



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

After InnoDB writes to the redo log, it writes the effects of those change events to the database pages. It takes a copy of the unmodified database pages before this and records the unmodified page to the undo log.

Undo Tablespaces

- By default, undo logs reside in two undo tablespaces.
 - Having two undo tablespaces reduces the maximum size of each.
 - They reside in the MySQL data directory by default.
 - Change their location by setting the `innodb_undo_directory` option.
- Undo logs have different I/O patterns than standard data.
 - These patterns make them well suited to storing on SSD.
- Set the `innodb_rollback_segments` option to change the number of rollback segments allocated to each undo tablespace.
 - The default value of 128 is the maximum number allowed.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Temporary Table Undo Log

- The undo logs for temporary tables are handled differently from other tables.
- Temporary tables do not require redo logs.
 - After a crash, InnoDB does not need to rebuild any temporary tables.
- The undo logs for temporary tables are stored in the rollback segments of the global temporary tablespace file (`ibtmp1`).



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Temporary table undo logs reside in the global temporary tablespace and are used for user-created temporary tables and related objects. Temporary table undo logs are not redo-logged, because they are not required for crash recovery. They are used only for rollback while the server is running. This special type of undo log benefits performance because it reduces the I/O overhead of redo logging.

Quiz



Which of the following cannot be a target for storing a table when you execute a CREATE TABLE ... TABLESPACE statement?

- a. File-per-table tablespace
- b. General tablespace
- c. System tablespace
- d. Undo tablespace



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Answer: d

Topics

- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- How MySQL Stores Data
- How MySQL Stores Metadata
- InnoDB Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin and Component Interface

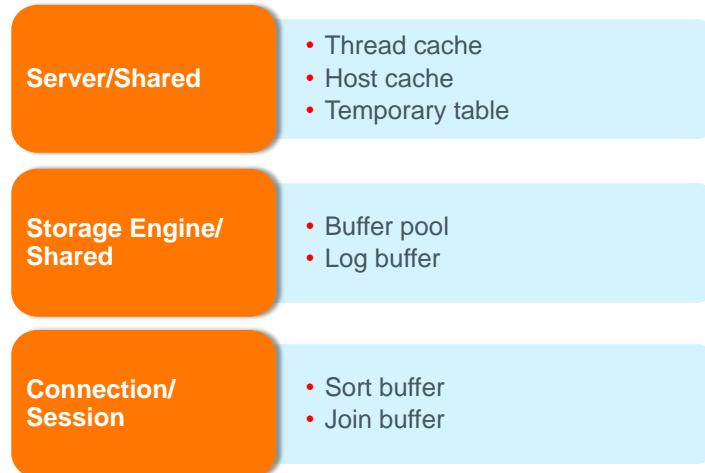


ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

How MySQL Uses Memory

- The MySQL server allocates memory in three different categories:



ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

If you use the MEMORY storage engine, MySQL uses the main memory as principal data store. Other storage engines can also use main memory for data storage, but MEMORY is unique for being designed to store data only in local memory.

How MySQL Uses Memory

- Global
 - Allocated when the server starts
 - Shared by the server process and its threads
 - Allocated by and for different components:
 - MySQL maintains its own instance memory.
 - InnoDB and NDB maintain internal memory stores.
- Session
 - Allocated for each thread
 - Dynamically allocated and deallocated
 - Used for handling query results
 - Buffer sizes are usually per session
 - For example: `sort_buffer_size`, `read_buffer_size`, and `binlog_cache_size`
 - `TempTable` has a shared global limit for all sessions (`temptable_max_ram`).



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Global Memory

- Allocated per MySQL server instance
- Allocated once when the server starts and freed when the server shuts down
 - This memory is shared across all sessions.
- When all the physical memory has been used up, the operating system starts swapping.
 - This has an adverse effect on MySQL server performance and can cause the server to crash.
- Specific buffers and caches include:
 - Grant table buffers for authorization
 - Storage engine buffers such as InnoDB's log buffer (`innodb_log_buffer_size`) and MyISAM's key cache (`key_buffer_size`)
 - Table open caches that hold descriptors for open tables (`table_open_cache`)



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

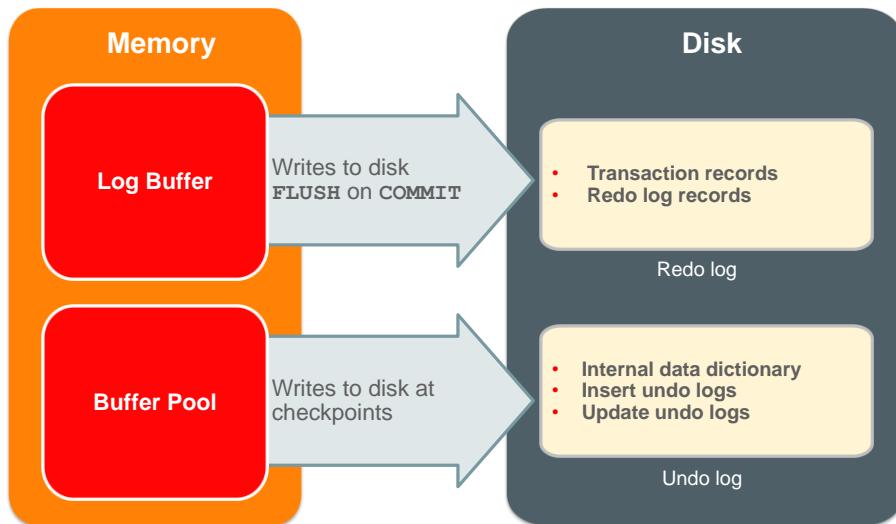
Session Memory

- Dynamically allocated per session
 - Per client connection or thread
- Freed when the session ends or is no longer needed
- Mostly used for handling query results
 - Some memory is dedicated to managing the connection buffer and thread stack.
 - Internal temporary tables use the `TempTable` (default) or `MEMORY` engine.
 - Most connection memory is used by its result buffer while executing statements.
- The sizes of the buffers used are per connection.
 - Example: A `sort_buffer_size` of 1 MB with 100 connections means that there could be a total of 100 MB used for all sort buffers simultaneously.
 - Some buffers can be allocated multiple instances per connection, for example, the join buffer and temporary table.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Log Files and Buffers



ORACLE

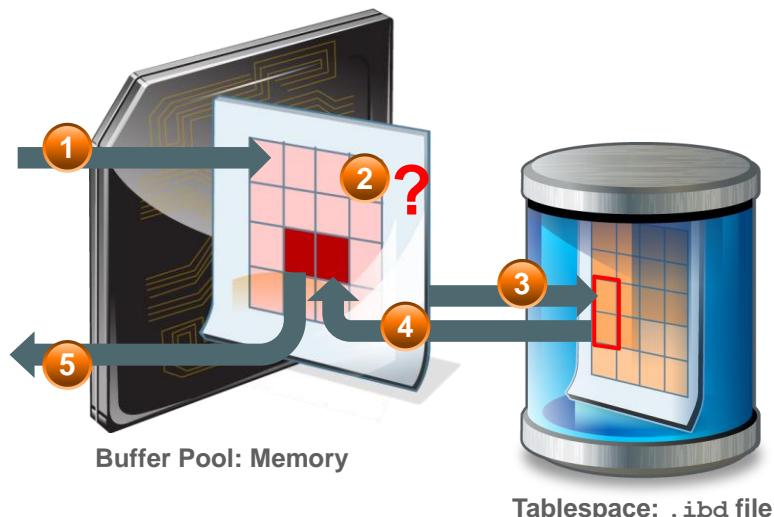
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Log Buffer and the Redo Log

As a client performs a transaction, it writes data change operations to the log buffer in memory. InnoDB writes the buffered log information to the redo log files on disk when the transaction commits, although you can configure this to happen less frequently.

If a crash occurs while the tables are being modified, the redo log files are used for automatic recovery. When the MySQL server restarts, it reapplies the changes recorded in the logs to ensure that the tables reflect all committed transactions.

InnoDB Buffer Pool



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

InnoDB maintains one or more buffer pools that cache frequently used data and indexes in main memory.

When you execute a read query from a client program, the following operations occur:

1. The client submits the request.
2. InnoDB checks to see if the data pages that it needs are in the buffer pool.
3. If the required data pages are not in the buffer pool, InnoDB requests the data from the tablespace.
4. InnoDB puts the data pages in the buffer pool.
5. MySQL returns the results to the client.

Configuring the Buffer Pool

- Assign as much RAM as you can to the buffer pool to avoid disk I/O on hot data
 - Set the value of `innodb_buffer_pool_size` so that it uses 70–80% of memory.
 - On a host that is dedicated to using MySQL:
 - Calculate the RAM used by the operating system and occasional administrative programs such as backups that minimize paging
 - Assign 50–80% of the remaining memory to the buffer pool depending on the session memory requirements
 - Example: On a 16 GB Linux system dedicated to MySQL, assign approximately 12 GB to the buffer pool.
- Enable multiple buffer pools to minimize mutex contention
 - InnoDB automatically configures eight buffer pool instances when your total buffer pool size is larger than 1 GB.
 - Set `innodb_buffer_pool_instances` so that each instance uses at least 1 GB.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

BufferPool size allocation

Assuming InnoDB is used for all or most of the tables, you may allocate the following percentage of remaining memory to InnoDB buffer pool as a starting point:

- 70–80% for default `max_connections` of 151
- 60–70% for `max_connections` of 1000
- 50% for `max_connections` of 5000

Monitor the memory usage and adjust the memory allocation as necessary to prevent the OS from swapping the memory to disk.

Buffer Pool Mutexes and Contention

A *mutex* is a mechanism that ensures mutual exclusion of memory or some other resource. InnoDB uses a single mutex for each buffer pool to protect against issues with concurrent access.

If each buffer pool instance manages a large number of pages, those pages cannot be served in parallel to multiple clients, because a single mutex protects that instance. If you have multiple buffer pool instances, each instance is protected by its own mutex. This means that you improve InnoDB's concurrency with large numbers of connections and a large buffer pool size when you have multiple buffer pool instances.

Topics

- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- How MySQL Stores Data
- How MySQL Stores Metadata
- InnoDB Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin and Component Interface



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL Plugin Interface

- Daemon plugins are run by the server.
- The plugin API allows loading and unloading of server components.
 - Supports dynamic loading, without restarting server
- Example plugins:
 - Japanese MeCab full-text parser
 - PAM Authentication
 - Thread Pool plugin
 - Rewriter plugin
 - Third-party storage engines
- MySQL supports both client and server plugins.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The plugin API supports:

- Full-text parser plugins that can be used to replace or augment the built-in full-text parser.
 - For example, a plugin can parse text into words by using rules that differ from those used by the built-in parser. This is useful to parse text with characteristics different from those expected by the built-in parser. The MeCab plugin parses Japanese text into meaningful words.
- Authentication plugins that enable authentication by external services such as PAM or LDAP
- Query rewriter plugins that rewrite statements before or after they are parsed

The plugin interface requires the `plugin` table in the `mysql` database. This table is created as part of the MySQL installation process.

MySQL Component Interface

- Both components and plugins can dynamically extend the functionality of MySQL server.
- MySQL component interface is introduced to overcome some of the architectural constraints of the plugin interface.
 - Components are self-contained code containers that interact with other code exclusively by implementing and consuming services via the registry.
 - Each component can communicate with other components only through services.
- Example components:
 - Error log components
 - Password Validation component
 - Audit Message component



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The component subsystem is designed to overcome some of the architectural issues of the plugin subsystem, namely:

- Plugins can only "talk" to the server and not with other plugins.
- Plugins have access to the server symbols and can call them directly; that is no encapsulation.
- There's no explicit set of dependencies of a plugin; thus, it's hard to initialize them properly.
- Plugins require a running server to operate.

Summary



In this lesson, you should have learned how to:

- Explain how MySQL processes, stores, and transmits data
- Describe the Transactional Data Dictionary
- Configure InnoDB tablespaces
- Explain how MySQL uses memory
- Configure the InnoDB buffer pool
- List some of the available plugins



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Practices

- 3-1: Configuring Tablespaces
- 3-2: Configuring the Buffer Pool



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

4

Configuring MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Objectives



After completing this lesson, you should be able to:

- Configure MySQL servers by using option files and command-line options
- Configure the `mysql` client
- Change MySQL settings dynamically
- Launch multiple MySQL servers on the same host



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

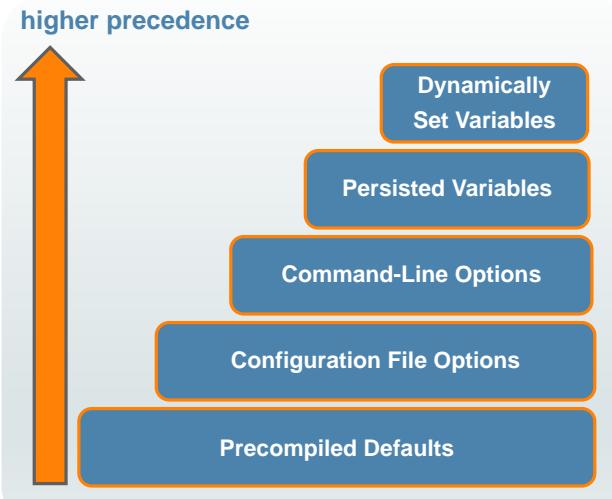
- Server Options, Variables, and the Command Line
 - Option Files
 - System Variables
 - Launching Multiple Servers on the Same Host



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL Configuration Options



- When you start MySQL clients or the server, specify configuration options:
 - At the command line
 - In a configuration file
- If you do not specify a value for an option, MySQL uses a precompiled default.
- On a running server, you can change the values of *dynamic variables*.
- On a running server, you can set persisted variables that remain effective after the server restarts.

ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Configuration files are also called *option files*.

MySQL applies options in the following precedence (lowest to highest):

- Precompiled defaults
- Configuration file options
 - If you have multiple configuration files, options are applied from them in order, with later configuration files overriding options in earlier files.
- Command-line options
- Persisted variables configured on the server
- Dynamic variables that you set at run time

If you specify a value in a configuration file but specify a different value when you launch `mysqld` at the command line, the command-line option takes precedence. If you subsequently change the value by setting a dynamic variable, the option assumes that new value.

Deciding When to Use Options

The following list contains some of the many things you can achieve by setting option values:

- Control which log files the server writes
- Specify the locations of important directories and files
 - Data directory, log files, PID, and socket files
- Override the server's built-in values for performance-related variables
 - The maximum number of simultaneous connections
 - Sizes of buffers and caches
- Enable or disable precompiled storage engines at server startup



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Displaying Configured Server Options

To find out what options your server supports, execute one of the following commands:

- At the mysql prompt:

```
SHOW GLOBAL VARIABLES;
```

- At the command line, if the server is running:

```
mysqladmin variables
```

- The preceding commands also show values that you changed dynamically after starting MySQL.

- At the command line, even if the server is not running:

```
mysqld --verbose --help
```

- The values include precompiled defaults, option file settings, and command-line options. Persisted variables are excluded.
- Unlike other variants of the mysqld command, this command does not start the mysqld process.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Option Naming Convention

In general, option names have the following conventions:

- **Option files:** Lowercase option names with words separated by the dash “-” or underscore “_” character.
- **Command line:** Same as option file but prefixed with two dashes “--”
- **Variable (in the running server):** Same as option file but with words always separated by the underscore character “_”

Examples:

Option file	Command line	Variable
datadir	--datadir	datadir
log-error	--log-error	log_error
default_password_lifetime	--default_password_lifetime	default_password_lifetime



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

If the command-line variant of an option uses underscores, the option file variant also uses underscores.

Using Command-Line Options

- Launch `mysqld` at the command prompt, providing command-line options:

```
mysqld --no-defaults --basedir=/opt/mysql --datadir=/mysql/data  
--user=mysql --pid-file=/mysql/pid --socket=/mysql/socket --port=3307
```

- Create scripts containing invocations that you use frequently so that you can avoid typing out long command lines.
- The `mysqld_safe` script launches `mysqld` with command-line options.

- Launch command-line clients:

- `mysql`

```
mysql  
mysql --socket=/mysql/socket -uroot -p
```

- `mysqladmin`

```
mysqladmin -uroot -p variables
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The example `mysqld` command shown in the slide launches a MySQL server process even on hosts that have a running `mysqld` with default settings.

The `--no-defaults` option causes `mysqld` to disregard any settings that are in option files. This means that the only options that apply to the newly started server are precompiled defaults and command-line options. This option is useful when you want to launch ad hoc test or development instances of MySQL on the same host as a running server, without reading the existing option files.

Note: `mysqld_safe` is not installed with RPM package on Linux running systemd.

Topics

- Server Options, Variables, and the Command Line
- **Option Files**
- System Variables
- Launching Multiple Servers on the Same Host



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Reasons to Use Option Files

- You do not need to specify options on the command line each time you start the server.
 - More convenient
 - Less error-prone for complex options
- You can review an option file to view the server configuration in one place.
- You can create multiple configurations with grouped options, each in its own configuration file.
 - Start multiple servers on the same host with different configurations.
 - Start test or development servers with alternative configurations.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Option File Locations

- MySQL server looks for files in standard locations.
- Standard option file names are different for Linux and Windows:
 - In Linux, use the `my.cnf` file.
 - In Windows, use the `my.ini` file.
 - MySQL also recognizes `my.cnf`, but `my.ini` is standard.
- The server may read multiple option files from different locations.
 - Most installations use a single option file for simplicity.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Option Files That Each Program Reads

- Run the MySQL program with the **--help** command-line option.
 - Server: Add **--verbose** to view server option files.

```
mysql --help --verbose
```

- Clients:

```
mysql -help  
mysqlslap --help
```

- Example output:

```
...  
Default options are read from the following files in the given order:  
/etc/my.cnf /etc/mysql/my.cnf /usr/local/mysql/etc/my.cnf ~/.my.cnf  
...
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The following command filters the output of the command:

```
mysql --help --verbose 2> /dev/null | grep -A1 "Default options"
```

If you launch MySQL from a script or service helper instead of executing the `mysql` program directly, that script or helper might read additional option files.

Standard Option Files

- Linux:
 - The file **/etc/my.cnf** serves as a global option file used by all users.
 - Create user-specific option files named **.my.cnf** in the user's home directory.
 - If the **MYSQL_HOME** environment variable is set, it searches for the **\$MYSQL_HOME/my.cnf** file.
- Windows:
 - The global option files are **my.ini** and **my.cnf** in any of the following directories:
 - %PROGRAMDATA%\MySQL\MySQL Server 8.0: where %PROGRAMDATA% is usually C:\ProgramData
 - The MySQL base installation directory, typically: C:\Program Files\MySQL\MySQL 8.0 Server
 - %WINDIR%: usually C:\Windows
 - C:\



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Option File Groups

Options in option files are organized into groups.

- Each group is preceded by a `[group-name]` line that names the group.
- Typically, the group name is the category or name of the program to which the group of options applies.

Option File

All Clients

- `[client]`

Client-Specific

- `[mysql]`
- `[mysqldump]`
- `...`

All Servers

- `[server]`

Server-Specific

- `[mysqld]`
- `[mysqld-8.0]`
- `...`



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Option Groups That Each Program Reads

- Run the MySQL program with the `--help` command-line option.
 - Option groups appear below the option file locations.
 - Server: Add `--verbose` to view server option files.
- Examples:

```
mysql> mysqld --help --verbose | grep "following groups"
The following groups are read: mysqld server mysqld-8.0
```

```
mysql> mysql --help | grep "following groups"
The following groups are read: mysql client
```

```
mysql> mysqladmin --help | grep "following groups"
The following groups are read: mysqladmin client
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Option Group Names

Examples of groups include:

- **[client]**: Options that apply to all client programs
 - Often used to specify connection parameters common to all clients
- **[mysql]** and **[mysqldump]**: Options that apply to mysql and mysqldump clients, respectively
 - Other client programs read option groups with their own names.
- **[server]**: Options that apply to all server programs or scripts
- **[mysqld]**, **[mysqld-8.0]**, and **[mysqld_safe]**: Options that apply to specific server programs or scripts



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Client Options: Examples

Some examples of mysql client options:

- **user** and **password**: Plain text authentication credentials
 - Use `mysql_config_editor` to create encrypted credentials.
 - Creates a `.mylogin.cnf` file in the home directory to store the credentials
- **database=dbname**: Specify the database or schema to use
- **prompt=prompt**: Replaces the default `mysql>` prompt
 - Example: `prompt='\\u@\\h[\\d]> '`
 - Produces the prompt `username@hostname [database] >`
- **safe-updates**: Prevents UPDATE or DELETE operations that do not specify a WHERE clause
- **init-command=SQL**: SQL string that the client executes as soon as it connects
- **tee=filename**: Appends all commands and output to the specified file



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Writing Option Files

- To create or modify an option file:
 - The editing user must have write permission on the file.
 - MySQL programs need only read access.
 - Server and client programs read option files but do not create or modify them.
 - MySQL programs ignore any world writable option files.
- To write an option in an option file:
 - Use the option file form:
 - Similar to the command-line form, but omitting the leading dashes
 - If an option takes a value, spaces are allowed around the equal (=) sign.
 - This is not true for options specified on the command line, which do not accept spaces around the equal sign.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Option File Contents: Example

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock

log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

[client]
host=myhost.example.com
compress

[mysql]
show-warnings
prompt='\u@\[ \d]> '
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, note the following:

- [mysqld]: Options in this group apply to all versions of mysqld.
- [client]: Options in this group apply to all standard clients.
 - host: Specifies the server host name to which the client connects unless overridden by a command-line option
 - compress: Directs the client/server protocol to use compression for traffic sent over the network
- [mysql]: Options in this group apply only to the mysql client.
 - show-warnings: Tells MySQL to show any current warnings after each statement
 - prompt: Changes the prompt
- The mysql client uses options from both the [client] and [mysql] groups, so it would use all of the options shown under those headings.

Option Precedence in Option Files

- MySQL programs read configuration files in a defined order.
 - Viewed with *program* --help [--verbose]
- MySQL programs read options from multiple option groups.
 - Example: mysqld reads from the [mysqld], [server], and [mysqld-8.0] groups.
- If an option value is specified in multiple configuration files, options in later files override options in preceding files.
- If an option value is specified in multiple groups within the same file, options that are later in the file take precedence.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Loading or Ignoring Option Files from the Command Line

Specify one of the following as the **first option** on the command line to control which option files MySQL reads:

- **--no-defaults**: Do not read any option files.
- **--defaults-file=*file_name***: Use only the option file at the specified location.
- **--defaults-extra-file=*file_name***: Use an additional option file at the specified location after reading all the standard option files.
- Example:

```
mysql --defaults-file=/etc/my-opt.cnf
```

- Uses only the /etc/my-opt.cnf file and ignores the standard option files



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Loading Option Files with Directives

Place these directives in an option file so that programs load additional option files.

- **`!include file_name`**
 - Load additional options from `file_name`.
- **`!includedir directory`:**
 - Load additional options from files in the directory that end in `.cnf` (or `.ini` for Windows).
 - MySQL does not read the option files in the directory in a predictable order.
 - Ensure that there are no conflicting option values in the included files.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Displaying Options from Option Files

Execute `my_print_defaults` to display options per group from the command line.

- To display options in the [mysql] and [client] groups:

```
$ my_print_defaults mysql client  
--user=myusername  
--password=secret  
--host=localhost  
--port=3306
```

- Or (for the same result):

```
mysql --print-defaults mysql client
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Option Files Used by `my_print_defaults`

As with other MySQL client programs, `my_print_defaults` reads options from a standard set of option files.

- To view the set of option files, execute `my_print_defaults --help`.
- To modify the set of option files that `my_print_defaults` reads, use the command-line options shown in the section titled “Loading or Ignoring Option Files from the Command Line.”

Quiz



In addition to the [mysqldump] group, the mysqldump program reads options from which of the following option groups?

- a. [client]
- b. [mysql]
- c. [mysqld]
- d. [server]



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Answer: a

Topics

- Server Options, Variables, and the Command Line
- Option Files
- **System Variables**
- Launching Multiple Servers on the Same Host



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Server System Variables

- In a running server, configured options are known as *system variables*.
 - Variables that you have not configured in option files or in command-line options assume their precompiled default values.
- You can change the value of some system variables in a running server.
 - These are called *dynamic variables*.
- You can refer to system variable values in expressions or by querying Performance Schema tables.
 - `global_variables`: Global system variables
 - `session_variables`: System variables for the current session
 - `variables_by_thread`: Session system variables for each active session
 - `persisted_variables`: Persisted global system variable
 - `variables_info`: The source from which each system variable was most recently set.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The Performance Schema enables you to inspect internal execution of the server at run time.

Note: This course covers using Performance Schema in the lesson titled “Monitoring MySQL.”

System Variable Scope: GLOBAL and SESSION

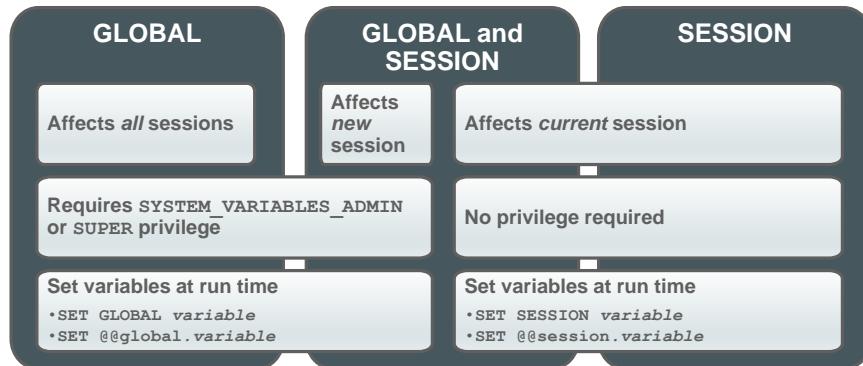
- MySQL maintains two scopes that contain system variables.
 - **GLOBAL** variables affect the overall operation of the server.
 - Changed with `SET GLOBAL variable_name` or `SET @@global.variable_name`
 - **SESSION** variables affect individual client connections.
 - Changed with `SET SESSION variable_name` or `SET @@session.variable_name`
- Variables are global, session, or both.
 - Variables that exist in both scopes have global and per-connection values that you can set independently.
- Examples of variables and their scope include:
 - Global only: `innodb_buffer_pool_size`, `max_connections`
 - Both global and session: `sort_buffer_size`, `join_buffer_size`
 - Session only: `timestamp`, `error_count`



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Dynamic System Variables

- Change dynamic variables at run time to avoid changing the option files and restarting the server.
- If you change a variable with both global and session scope:
 - Changing the global variable affects all new connections
 - Changing the session variable affects the current connection



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Changing Variable Values

- Setting a global variable requires SYSTEM_VARIABLES_ADMIN or SUPER privilege.
- Setting a session variable requires no special privilege.
 - A client can change only its own session variables.

```
SET GLOBAL max_connections=200;  
SET @@session.sort_buffer_size=512*1024;
```

- LOCAL and @@local are synonyms for SESSION and @@session.
- If you do not specify GLOBAL or SESSION:
 - SET changes the session variable if it exists
 - SET produces an error if the session variable does not exist

```
mysql> SET max_connections=200;  
ERROR 1229 (HY000): Variable 'max_connections' is a GLOBAL variable and should be  
set with SET GLOBAL
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

SUPER privilege is deprecated and will be removed in a future version of MySQL. It is recommended to use the appropriate **dynamic privileges** that have more limited scope for better security. For example, grant SYSTEM_VARIABLES_ADMIN to users who need to change global variables instead of SUPER privilege.

Persisting Global Variables

Use `SET PERSIST variable_name = value` to maintain global variable values across server restarts.

- Use `DEFAULT` in place of `value` to restore the default value.
- Requires the following server privileges:
 - `SYSTEM_VARIABLES_ADMIN`
 - `PERSIST_RO_VARIABLES_ADMIN`
- Stores details of changes in the `mysqld-auto.cnf` file in the data directory, in JSON format:
 - The variable name and current value
 - When and by whom the change was made

```
# cat /var/lib/mysql/mysqld-auto.cnf
{
  "Version":1,
  "mysql_server": {
    "max_connections": {
      "Value":"152",
      "Metadata": {
        "Timestamp":1526635140519175,
        "User":"root",
        "Host":"localhost"
      }
    }
  }
}
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Persisted system variables are processed at the end of server startup and have higher precedence compared to option files and command-line options.

The `SET PERSIST_ONLY` statement persists the global variable to the file `mysqld-auto.cnf` without setting its runtime value. This allows you to set the read-only system variables. The value will take effect after a server restart.

The `RESET PERSIST` statement can be used to remove the persisted variables from the MySQL server. It can remove all the persisted variables or a single persisted variable. The syntax is :

- `RESET PERSIST [[IF EXISTS] system_var_name]`

Some variables cannot be persisted, and some variables are persist-restricted. See <https://dev.mysql.com/doc/refman/8.0/en/nonpersistible-system-variables.html> for the full list.

Persist-restricted variables can be set as `PERSIST_ONLY` by a user connected to the MySQL server using an SSL certificate X.509 Subject value specified in the `persist_only_admin_x509_subject` server variable.

Displaying System Variables

- List all available variables and their values:

```
SHOW [GLOBAL|SESSION] VARIABLES;
```

- List specific variable values:

```
mysql> SHOW VARIABLES LIKE 'read_only';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| read_only     | OFF   |
+-----+-----+
```

- Set a new value and then display it:

```
mysql> SET GLOBAL read_only=ON;
mysql> SHOW VARIABLES LIKE 'read_only';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| read_only     | ON    |
+-----+-----+
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

read_only is a GLOBAL variable; the session value is always the same as the global value. Also, it requires the SYSTEM_VARIABLES_ADMIN or SUPER privilege to modify the value.

The example in the slide uses the `LIKE` operator and the string '`read_only`' to find the variable. Use the `LIKE` operator to specify any variable by using the whole variable name or by entering a partial variable name with wildcards that include the underscore (`_`), which matches single characters, or the percent sign (`%`), which matches any number of characters.

Viewing Variables with Performance Schema

- The `global_variables`, `session_variables`, and `persisted_variables` tables contain global variables, current session variables, and persisted global variables, respectively.

```
mysql> SELECT * FROM global_variables WHERE VARIABLE_NAME='pid_file';
+-----+-----+
| VARIABLE_NAME | VARIABLE_VALUE |
+-----+-----+
| pid_file      | /var/run/mysqld/mysqld.pid |
+-----+-----+

mysql> SELECT * FROM persisted_variables;
+-----+-----+
| VARIABLE_NAME | VARIABLE_VALUE |
+-----+-----+
| max_connections | 152 |
+-----+-----+
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Viewing Variables with Performance Schema

- The **variables_by_thread** table contains session variables for all active threads.
- The **variables_info** table contains the source from which each system variable was most recently set and its range of values.

```
mysql> SELECT * FROM variables_by_thread
      -> WHERE THREAD_ID=27 AND VARIABLE_NAME='sort_buffer_size';
+-----+-----+-----+
| THREAD_ID | VARIABLE_NAME | VARIABLE_VALUE |
+-----+-----+-----+
|     27 | sort_buffer_size | 262144        |
+-----+-----+-----+

mysql> SELECT VARIABLE_NAME, VARIABLE_SOURCE, VARIABLE_PATH FROM variables_info
      -> WHERE VARIABLE_NAME='datadir';
+-----+-----+-----+
| VARIABLE_NAME | VARIABLE_SOURCE | VARIABLE_PATH  |
+-----+-----+-----+
| datadir       | GLOBAL          | /etc/my.cnf   |
+-----+-----+-----+
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The **variables_info** table contains these columns:

- **VARIABLE_NAME**: The variable name
- **VARIABLE_SOURCE**: The source from which the variable was most recently set. Possible values are COMMAND_LINE, COMPILED, DYNAMIC, EXPLICIT, EXTRA, GLOBAL, LOGIN, PERSISTED, SERVER, and USER.
- **VARIABLE_PATH**: If the variable was set from an option file, this is the path name of the file.
- **MIN_VALUE** and **MAX_VALUE**: The minimum and maximum permitted values for the variable
- **SET_TIME**: The time at which the variable was most recently set
- **SET_USER** and **SET_HOST**: The user name and host name of the client user that most recently set the variable.

Topics

- Server Options, Variables, and the Command Line
- Option Files
- System Variables
- Launching Multiple Servers on the Same Host



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Launching Multiple Servers on the Same Host

- Useful for many administrative purposes, such as:
 - Testing a new release of MySQL
 - Testing replication and high availability
 - Partitioning client groups into different servers
- Can be achieved with multiple methods:
 - Start `mysqld` or `mysqld_safe` using command-line options.
 - Start `mysqld` or `mysqld_safe` with a different option file for each server.
 - Use the `--defaults-file` option.
 - `mysqld_multi` manages multiple similar servers with different settings.
 - `systemd` service manager can manage multiple service instances.
- Servers must not share filesystem or network resources with other servers.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Run multiple servers when you want to test a new release of MySQL on the same machine where the production server is running.

Command-line options example:

```
mysqld --socket=/mysql/socket2 --port=3312 --datadir=/mysql/data2  
mysqld --socket=/mysql/socket3 --port=3313 --datadir=/mysql/data3
```

Option file example:

1. Create an option file for each server, for example `/mysql/my.cnf2`:

```
[mysqld]  
socket=/mysql/socket2  
port=3312  
datadir=/mysql/data2
```

2. Start MySQL server instance:

```
mysqld_safe --defaults-file=/mysql/my.cnf2
```

Note: `mysqld_safe` and `mysqld_multi` are not installed with RPM package on Linux running systemd.

Settings That Must Be Unique

- **Data Directory**
 - Start each server with a unique value for the `--datadir` option.
- **Connection Layer**
 - Specify unique connection parameters by starting each server with a unique value for the `--port` (or `--bind-address`), `--socket`, and `--shared-memory-basename` options.
- **Log and PID Files**
 - By default, these are in the data directory, which must be unique for each server.
 - If you use nondefault locations, specify unique values for `--log-error` and other log file options and the `--pid-file` option.
- **InnoDB Tablespaces and Log Files**
 - By default, these are in the data directory.
 - If you use nondefault locations, specify unique values for tablespaces.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Only one instance of MySQL can listen on each TCP/IP port, which is a combination of TCP port and IP address. You can run two instances of MySQL on the same port number on the same host if they listen on different IP addresses. For example, you can have two instances on port 3306 if one has a `bind-address` of 127.0.0.1 and the other a `bind-address` of 192.168.1.14.

Note: The MySQL log files are described in more detail in the lesson titled “Monitoring MySQL.”

`mysqld_multi`

- Designed to manage several `mysqld` processes on the same host
 - Each `mysqld` process listens for connections on a unique UNIX socket file, TCP/IP port, named pipe, or shared memory base name.
- Applies options to each server N from groups named `[mysqld N]` in configuration files
 - Each group contains options that apply to a single numbered host.
 - Examples: `[mysqld1]`, `[mysqld3]`
 - Specify configuration files in the usual way.
 - Standard configuration files including `/etc/my.cnf`
 - Files that you specify with `--defaults-file` or `--defaults-extra-file`
- Start two `mysqld` instances that:
 - Apply settings from only `multi.cnf`
 - Read the sections `[mysqld1]` and `[mysqld3]`, respectively

```
mysqld_multi --defaults-file=multi.cnf start 1,3
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

For some Linux platforms, MySQL installation from RPM or Debian packages includes `systemd` support for managing MySQL server startup and shutdown. On these platforms, `mysqld_multi` is not installed because it is unnecessary.

mysqld_multi: Example Configuration File

```
[mysqld1]
user=mysql
datadir=/mysql/data1
port=3311
socket=/mysql/socket1

[mysqld2]
user=mysql
datadir=/mysql/data2
port=3312
socket=/mysql/socket2

[mysqld3]
user=mysql
datadir=/mysql/data3
port=3313
socket=/mysql/socket3
```

Each server instance has a unique data directory, port, and socket.

The user setting has a nondefault value, but it does not need to be unique.

The [mysqld3] option group contains the settings applied by the third of the servers managed by mysqld_multi.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Any options that you do not specify in the configuration file assume their default values. In some cases, the default values are file paths relative to the data directory and are effectively unique. For example, the default location of the PID file is in the data directory of the MySQL server, so all three servers in the configuration file have a unique PID file even though it is not explicitly configured.

systemd: Multiple MySQL Servers

- The `systemd` service manager uses the `mysqld@.service` configuration file to manage multiple MySQL server instances.
- When you start a MySQL server instance using:
`systemctl start mysqld@replica01`
 - `replica01` represents the instance identifier.
 - `systemd` starts `mysqld` with the `--defaults-group-suffix=@%I` option where `%I` is substituted with `replica01`.
 - In addition to the `[mysqld]` and `[server]` groups, `mysqld` reads options from the `[mysqld@replica01]` group.
- For every MySQL server instance, add an option group `[mysqld@<unique_id>]` into the option file. Specify all the instance-specific options into the option group.
- Check the status of multiple instances by using a wildcard:
`systemctl status mysqld@replica*`



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

You use `systemd` to deploy multiple servers in the practice titled “Running Multiple MySQL Servers on the Same Host with `systemd`.”

An example option file for two instances:

```
[mysqld@replica01]
datadir=/var/lib/mysql-replica01
socket=/var/lib/mysql-replica01/mysql.sock
port=3307
log-error=/var/log/mysqld-replica01.log

[mysqld@replica02]
datadir=/var/lib/mysql-replica02
socket=/var/lib/mysql-replica02/mysql.sock
port=3308
log-error=/var/log/mysqld-replica02.log
```

Quiz



The `sql_notes` variable is both global and session and specifies if `Note` level events are logged as warnings. If you change its value from 1 (the default) to 0 with the following command, what is the effect?

```
SET GLOBAL sql_notes=0;
```

- a. All connections stop logging `Note` events.
- b. Existing connections continue to log `Note` events; new connections do not.
- c. Existing connections that are currently logging `Note` events continue doing so. All other connections including new connections stop logging them.
- d. New connections are prevented from executing the command `SET SESSION sql_notes=1`.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary



In this lesson, you should have learned how to:

- Configure MySQL servers by using option files and command-line options
- Configure the `mysql` client
- Change MySQL settings dynamically
- Launch multiple MySQL servers on the same host



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Practices

- 4-1: Modifying a Setting by Using Command-Line Arguments
- 4-2: Modifying the Configuration File
- 4-3: Changing Dynamic Settings
- 4-4: Persisting Global Variables
- 4-5: Configuring the Client
- 4-6: Running Multiple MySQL Servers on the Same Host with `systemd`



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

5

Monitoring MySQL



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Objectives



After completing this lesson, you should be able to:

- Configure and view MySQL log files
- Identify slow queries
- Configure MySQL Enterprise Audit
- Use MySQL variables to monitor activity in MySQL
- Use MySQL Enterprise Monitor to view activity in MySQL



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

- Monitoring MySQL with Log Files
- Monitoring MySQL with Status Variables
- Monitoring MySQL with Performance Schema
- MySQL Enterprise Audit
- MySQL Enterprise Monitor
- Monitoring User Activity



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Monitoring MySQL with Log Files

MySQL uses several types of logs to record information about server activity.

- **Error log:** Diagnostic messages regarding startups, shutdowns, and abnormal conditions
- **Binary log:** Data modifications
- **General query log:** All statements that the server receives from clients
- **Slow query log:** Queries that take a long time to execute
- **Audit log:** Rule-based monitoring, logging, and blocking of connection and query activity; available with Enterprise Edition



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Use these logs to assess the operational state of the server, for data recovery after a crash, for replication purposes, to help determine which queries are running slowly, and for security and regulatory compliance.

Error log is enabled by default, and it is written to a file or console (stderr). Binary log is enabled by default with MySQL 8.0; it was disabled by default prior to MySQL 8.0. Other log files are disabled by default.

It is important to understand that log files, particularly the general query log, can grow to be quite large.

Log File Characteristics

- Can use large amounts of disk space
- Can be stored in files
- Can be stored in tables
 - General and slow query logs only
- Can be encrypted
 - Audit and binary logs only
- Are written in text format
 - Except binary log



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Error Log

- Is enabled by default
- Records the diagnostic messages such as errors, warnings, and notes that occur during server startup and shutdown and while the server is running
- Uses the MySQL component architecture starting from MySQL 8.0. Consists of:
 - Filter components
 - `log_filter_internal`: Filters based on log event priority and error code (default)
 - `log_filter_dragnet`: Filters based on user-supplied rules
 - Sink (writer) components
 - `log_sink_internal`: Built-in (default) component writes to a file or console (stderr)
 - `log_sink_json`: Writes in JSON format to a file or console (stderr)
 - `log_sink_syseventlog`: Writes to the system log of the Operating Systems
- Is configured using the `log_error_services` variable
 - Specify the filter component followed by one or more sink components
 - For example: `log_filter_internal; log_sink_internal; log_sink_json`

ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The `log_filter_internal` component uses the `log_error_verbosity` and `log_error_suppression_list` server variables to filter the error logs based on error event priority and error code.

The `log_filter_dragnet` component uses the `dragnet.log_error_filter_rules` system variable to define the filtering rules. See <https://dev.mysql.com/doc/refman/en/error-log-rule-based-filtering.html> for the syntax to specify the filtering rules.

The default error log destination for `log_sink_internal` and `log_sink_json` components:

- On Windows:
 - If `--console` option is enabled, writes to the console (stderr).
 - Else, if the `--log-error` is defined with a file name, writes to the specified file.
 - Else, if the `--pid-file` is defined with a file name, writes to file with a suffix of `.err` in the data directory.
 - Else, writes to `host_name.err` in the data directory.
- On Linux and Unix-like systems:
 - If the `--log-error` is defined with a file name, writes to the specified file.
 - If the `--log-error` is defined without a file name, writes to `host_name.err` in the data directory.
 - If the `--log-error` is not defined, writes to the console (stderr).

Binary Log

- Contains data and schema changes and their time stamps
- Enabled by default in MySQL 8.0, use `log_bin=OFF` to disable binary logging
 - The server uses the `log_bin_basename` option value as a base name, adding an increasing sequential numeric suffix to the base name as it creates new log files.
- Is used for point-in-time recovery from backup, full recovery from backup, and replication
- Rotates when:
 - MySQL server restarts
 - It reaches its maximum size as set by `max_binlog_size`
 - You issue a `FLUSH LOGS` or `FLUSH BINARY LOGS` statement
- Can be inspected in various ways:
 - Metadata: `SHOW BINARY LOGS`, `SHOW MASTER STATUS` statements
 - Contents: `SHOW BINLOG EVENTS` statement, `mysqlbinlog` client



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Purging of binary logs

The binary logs can be removed or purged by:

- Setting the `binlog_expire_logs_seconds` or `expire_logs_days` variables
 - The default is 30 days.
 - Removals happen at startup and when the binary log is flushed.
- Executing the `PURGE BINARY LOGS` statement. There are two variants of the statement.
 - `PURGE BINARY LOGS TO 'log_name'`
 - `PURGE BINARY LOGS BEFORE datetime_expr`

All binary log files required by a connected slave server will not be purged.

FLUSH LOGS in Replication Environments

By default, the MySQL server writes `FLUSH` commands to the binary log so that statements are replicated to the slaves. To prevent this from happening, use the optional `NO_WRITE_TO_BINLOG` keyword or its alias `LOCAL`.

Note: You can use the binary log as part of a backup strategy or for replicating change events from one server to another. For more information about using the binary log for:

- Backup and restore operations, see the lesson titled “Choosing a Backup Strategy”
- Replication, see the lesson titled “Configuring a Replication Topology”

General Query Log

- Is enabled by using the `general_log` server option
- Records connection information and details about statements received
 - Records the time and type of each connection and the process ID of all operations
 - Records all statements that are executed against all tables
 - Excludes update operations stored as row-based binary log on slave servers
- Grows very quickly
 - Enable it for short periods to gather a full record of all activities during those periods
- Has a large overhead
 - Not feasible to enable it for long periods on busy production systems



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

General Query Log: Example

- Contains event types and query contents
 - Example types: Query, Connect
- Shows details for each connection and query
 - Records the time of each connection and the process ID of all operations

```
date-and-time      12 Connect    root@localhost on  using Socket
date-and-time      12 Query       select @@version_comment limit 1
date-and-time      12 Query       SELECT DATABASE()
date-and-time      12 Init DB    world
date-and-time      12 Query       show databases
date-and-time      12 Query       show tables
date-and-time      12 Field List   city
date-and-time      12 Field List   country
date-and-time      12 Field List   countrylanguage
date-and-time      12 Query       SELECT * FROM city LIMIT 5
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Slow Query Log

- Is enabled by using the `slow_query_log` server option
- Records statements where the execution time exceeds a specified threshold
 - 10 seconds (by default)
 - Change this duration with the `long_query_time` server option.
 - Specify the number of seconds with microsecond precision.
 - Example: 0.03 is 30 milliseconds.
- Can log statements that do not use indexes, even those below `long_query_time`
 - Enable `log_queries_not_using_indexes`
- Can record additional statistics into `FILE` destination
 - Enable the `log_slow_extra` server option
 - Does not affect `TABLE` destination
- Is a text file that can be viewed directly
 - Can be summarized with the `mysqldumpslow` command-line program



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Slow Query Log: Logging Administrative and Replicated Statements

- The slow query log does not record administrative statements by default.
 - You can record such statements by enabling the `log_slow_admin_statements` server option.
- Statements replicated from a replication master do not appear in the slow query log by default, even if they exceed the time specified by the `long_query_time` server option.
 - To record such statements, enable the `log_slow_slave_statements` server option.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Filtering Slow Query Log Events

Additional server options that you can use to filter the slow query log output:

- `min_examined_row_limit`
 - It specifies the lowest number of rows that a statement must examine so that the slow query log records the statement.
- `log_throttle_queries_not_using_indexes`
 - It specifies the number of queries not using indexes within a 60-second period that the slow query log records.
 - After the slow query log records the number of those queries, it summarizes the number and aggregate time spent on the remaining statements in that time period.
 - By default, this server option has the value `0`, indicating that it records all such queries.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Slow Query Log: Example

Entries contain:

- Server date and time
- Information about the connection and query
- Time taken to run the query and hold locks

```
# Time: date-and-time
# User@Host: root[root] @ localhost []  Id:      9
# Query_time: 1.540755  Lock_time: 0.000079 Rows_sent: 354  Rows_examined:
2844401
SET timestamp=timestamp;
SELECT emp_no, salary FROM salaries WHERE from_date BETWEEN '1986-01-01' AND
'1986-01-07' ORDER BY from_date, salary;
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The slow query log records additional statistics when `log_slow_extra` is enabled:

```
# Time: date-and-time
# User@Host: root[root] @ localhost []  Id:      9
# Query_time: 1.588251  Lock_time: 0.000203 Rows_sent: 354
Rows_examined: 2844401 Thread_id: 9 Errno: 0 Killed: 0 Bytes_received: 0
Bytes_sent: 6057 Read_first: 1 Read_last: 0 Read_key: 1 Read_next: 0
Read_prev: 0 Read_rnd: 0 Read_rnd_next: 2844048 Sort_merge_passes: 0
Sort_range_count: 0 Sort_rows: 354 Sort_scan_count: 1
Created_tmp_disk_tables: 0 Created_tmp_tables: 0 Start: date-and-time
End: date-and-time
SET timestamp=timestamp;
SELECT emp_no, salary FROM salaries WHERE from_date BETWEEN '1986-01-01'
AND '1986-01-07' ORDER BY from_date, salary;
```

Viewing the Slow Query Log with mysqldumpslow

- The `mysqldumpslow` command-line program summarizes the contents of the slow query log.
- It groups similar queries together and:
 - Changes numeric parameters to N
 - Changes string parameters to ' S '
 - Shows the number of such queries and the average time and total time taken to run the queries
- Use the `-g` option to provide a search term.
 - Shows only summary information for statements that match the search term
- Use the `-s` option to sort the result.
 - Can be sorted on the average query time (default), average lock time, average rows sent, or count
 - Add the `-r` option to sort in reverse order



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

mysqldumpslow: Example

Search the slow query log for the text 'update `mem__inventory`.`MysqlServer`'

```
# mysqldumpslow -g 'update `mem__inventory`.`MysqlServer`' \
/var/lib/mysql/hostname-slow.log

Reading mysql slow query log from /var/lib/mysql/hostname-slow.log
Count: 558  Time=0.94s (524s)  Lock=0.00s (0s)  Rows=0.0 (0),
root[root]@localhost
  /* mem dbpool.default */ update `mem__inventory`.`MysqlServer` set
`timestamp`=N where hid=x'S

Count: 104  Time=0.93s (96s)  Lock=0.00s (0s)  Rows=0.0 (0),
root[root]@localhost
  /* mem dbpool.default */ update `mem__inventory`.`MysqlServer` set
`lastContact`=N, `hasLastContact`=N, `hasStartTime`=N, `timestamp`=N where
hid=x'S'
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Specifying TABLE or FILE Log Output

The `log_output` server option:

- Configures the destination of both the slow query log and the general query log
- Contains one or more of the values **FILE**, **TABLE**, or **NONE** (separated by commas)
 - **NONE**: The logs do not write to either the file or the table.
 - The presence of **NONE** causes **FILE** or **TABLE** to be ignored.
 - **FILE**: The logs write to the files specified by the `slow_query_log_file` and `general_log_file` MySQL server options, respectively.
 - These options have the default values `hostname-slow.log` and `hostname.log` in the MySQL data directory.
 - **TABLE**: The slow query log writes to the `slow_log` table and the general log writes to the `general_log` table, both in the `mysql` database.
 - The default value is **FILE**.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Log File Rotation

- Log files take up space over time. Periodically back up and remove old log files and recommence logging to new log files.
 - Use caution if you are using binary logs for replication.
- After backup, flush the logs. Flushing the logs:
 - Creates new binary log files
 - Closes and reopens the error log, general query log, and slow query log files
 - To create new logs, you must rename the current log files before flushing.
- Schedule log file rotation at regular intervals:
 - Create your own script or use the provided `mysql-log-rotate` script.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Flushing Logs

- After you back up the log files, flush the logs to force the MySQL server to start writing to new log files. Either:
 - Execute the `FLUSH LOGS` SQL statement or
 - Execute `mysqladmin flush-logs`
- Flushing the binary logs causes binary logging to recommence with the next file in the sequence.
- Flushing error log, general query log, and slow query log closes the log files, reopens them, and then recommences logging under the same file name.
 - To start new logs, rename the existing log files before flushing. Example:

```
# cd /var/lib/mysql
# mv server.log server.old
# mv mysql-slow.log mysql-slow.old
# mysqladmin flush-logs
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

You can flush individual type of log files using the log-specific `FLUSH` statements:

- `FLUSH BINARY LOGS`
- `FLUSH ERROR LOGS`
- `FLUSH GENERAL LOGS`
- `FLUSH SLOW LOGS`

Quiz



Which log is not stored in a human-readable text format?

- a. Binary log
- b. Error log
- c. General query log
- d. Slow query log



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Answer: a

Topics

- Monitoring MySQL with Log Files
- Monitoring MySQL with Status Variables
- Monitoring MySQL with Performance Schema
- MySQL Enterprise Audit
- MySQL Enterprise Monitor
- Monitoring User Activity

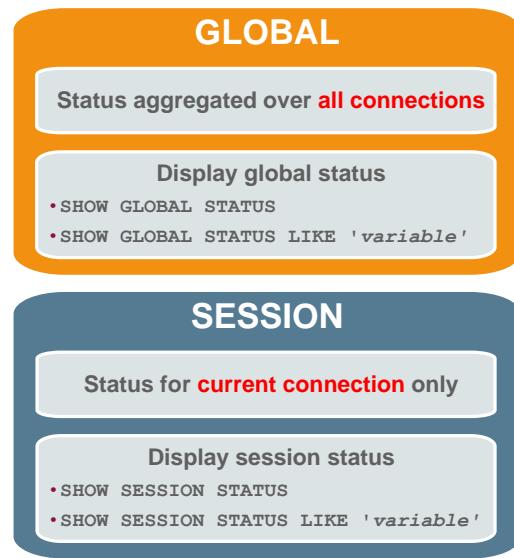


ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Status Variables

- Use the `SHOW STATUS` statement to assess the health of a system.
- Most of the status values are counters to indicate the number of times an event has occurred.
- Use a scope modifier (`GLOBAL` or `SESSION`) to display the global or local status information.
 - `LOCAL` is a synonym for `SESSION`.
 - If you do not specify a modifier, the default is `SESSION`.
- Some status variables have only a global value. For these, you get the same value for both `GLOBAL` and `SESSION`.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

As an example, the session value of the `Com_select` counts the number of `SELECT` statements that has executed in the current connection while the global value of the `Com_select` counts the total number of `SELECT` statements that has executed in all sessions.

Displaying Status Information

- Use `SHOW [GLOBAL | SESSION] STATUS`. For example:

```
mysql> SHOW GLOBAL STATUS;
+-----+-----+
| Variable_name      | Value |
+-----+-----+
| Aborted_clients    | 0     |
| Aborted_connects   | 0     |
| Binlog_cache_disk_use | 0     |
...
```

- Query the Performance Schema `global_status` or `session_status` tables. For example:

```
mysql> SELECT * FROM performance_schema.session_status;
+-----+-----+
| VARIABLE_NAME      | VARIABLE_VALUE |
+-----+-----+
| Aborted_clients    | 0             |
| Aborted_connects   | 1             |
| Binlog_cache_disk_use | 0             |
...
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

There are other Performance Schema tables that aggregate the status variables based on one or more grouping columns:

- `status_by_account`
- `status_by_host`
- `status_by_thread`
- `status_by_user`

Monitoring Status with mysqladmin

Use the `mysqladmin` command-line program with the following options to monitor MySQL:

- Display a short status message:

```
# mysqladmin status
Uptime: 240236 Threads: 5 Questions: 1683400 Slow queries: 3
Opens: 3440 Flush tables: 1 Open tables: 140
Queries per second avg: 7.007
```

- Display server status variables and their values:

```
# mysqladmin extended-status
```

- Equivalent to `SHOW GLOBAL STATUS`
- Use the `flush-status` option to clear status values.

- List active server threads:

```
# mysqladmin processlist --verbose
```

- Equivalent to `SHOW FULL PROCESSLIST`



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Combine the `mysqladmin` client program and shell options to generate useful outputs for monitoring. For example, the following command displays the difference between the current and previous values of all server status variables, every 100 seconds:

```
# mysqladmin extended -i100 --relative
```

Topics

- Monitoring MySQL with Log Files
- Monitoring MySQL with Status Variables
- **Monitoring MySQL with Performance Schema**
- MySQL Enterprise Audit
- MySQL Enterprise Monitor
- Monitoring User Activity



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Performance Schema

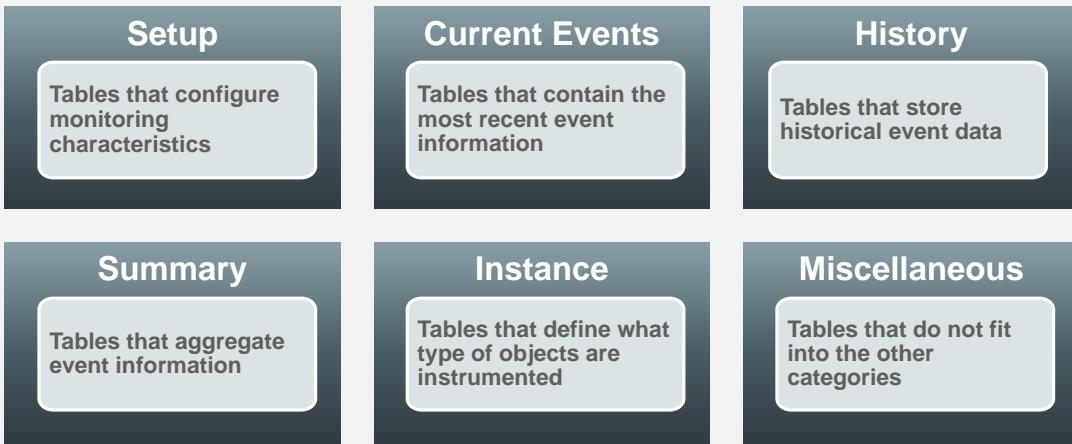
- A set of in-memory tables that MySQL uses to track performance metrics
 - Implemented as the PERFORMANCE_SCHEMA storage engine
 - Operates on tables in the `performance_schema` database
- Helps provide insight into database activity. For example:
 - Which queries are running
 - I/O wait statistics
 - Historical performance data
- Only available if support is configured during the build
 - Always available in Oracle binary distributions
 - If available, it is enabled by default.
 - To enable or disable it explicitly, start the server with the `performance_schema` variable set to an appropriate value.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Performance Schema Table Groups

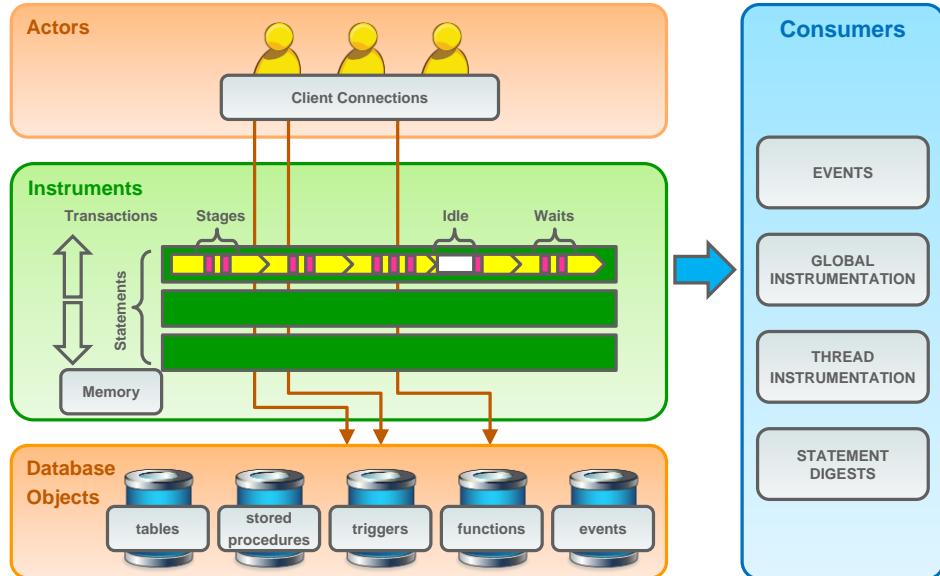


ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

There are many tables in the Performance Schema. They are loosely categorized as shown in the slide.

Configuring Performance Schema



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Performance Schema Setup Tables

You configure the Performance Schema by modifying the contents of the `setup_%` tables.

- **setup_actors**: Which foreground threads (client connections) are monitored
- **setup_objects**: Which database objects (tables, stored procedures, triggers, events) are monitored
- **setup_threads**: Which thread classes are instrumented
- **setup_instruments**: Which server metrics the Performance Schema collects
- **setup_consumers**: Where the instrumented events are stored



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Performance Schema Instruments

- Instruments correspond to instrumentation points within the MySQL Server source code.
- There are many instruments in the Performance Schema.
- An instrument name consists of a sequence of components separated by '/' characters.
Example names:
 - wait/io/file/myisam/log
 - wait/io/file/mysys/charset
 - wait/lock/table/sql/handler
 - wait/synch/mutex/sql/LOCK_delete
 - stage/sql/closing tables
 - stage/sql/Sorting result
 - statement/com/Execute
 - statement/sql/create_table
 - errors
- The components of an instrument from left to right go from general to more specific.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Top-Level Instrument Components

- **idle**: An instrumented idle event. This instrument has no subcomponents.
- **error**: An instrumented error event. This instrument has no subcomponents.
- **memory**: An instrumented memory event
- **stage**: An instrumented stage event
- **statement**: An instrumented statement event
- **transaction**: An instrumented transaction event. This instrument has no further components.
- **wait**: An instrumented wait event



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Accessing Performance Schema Metrics

Query Performance Schema tables just like any other table. For example, display the most recently monitored event (`events_waits_current` table):

```
mysql> SELECT * FROM events_waits_current\G
***** 1. row ****
    THREAD_ID: 0
    EVENT_ID: 5523
    EVENT_NAME: wait/synch/mutex/mysys/THR_LOCK::mutex
      SOURCE: thr_lock.c:525
    TIMER_START: 20160494489586
    TIMER_END: 20160494576112
    TIMER_WAIT: 86526
      SPINS: NULL
    OBJECT_SCHEMA: NULL
    OBJECT_NAME: NULL
    OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 142270668
    NESTING_EVENT_ID: NULL
      OPERATION: lock
    NUMBER_OF_BYTES: NULL
      FLAGS: 0
...
...
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The sys Schema

There are many instruments and tables in the Performance Schema. It can be difficult to know which ones to monitor.

The sys schema:

- Helps DBAs interpret the Performance Schema for typical tuning and diagnostic use cases
- Contains:
 - **Views:** Summarize Performance Schema data into an easier-to-understand format
 - **Stored procedures:** Assist DBAs in configuring Performance Schema and generating diagnostic reports
 - **Stored functions:** Query the Performance Schema configuration and format the output in different ways



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Using the sys Schema Example 1

Question: "Which user is consuming the most server resources?"

1. List the user summary views:

```
mysql> USE sys
Database changed
mysql> SHOW TABLES LIKE 'user%';
+-----+
| Tables_in_sys (user%)           |
+-----+
| user_summary                   |
| user_summary_by_file_io        |
| user_summary_by_file_io_type   |
| user_summary_by_stages         |
| user_summary_by_statement_latency |
| user_summary_by_statement_type  |
+-----+
6 rows in set (#.# sec)
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Using the sys Schema Example 1

2. The `user_summary` view shows an abnormally high number of file I/O events for the user Bob Smith:

```
mysql> SELECT * from user_summary\G
***** 1. row ****
      user: bobsmit
  statements: 5971
statement_latency: 1.20 m
statement_avg_latency: 12.04 ms
    table_scans: 193
        file_ios: 53784
file_io_latency: 15.35 m
current_connections: 1
  total_connections: 8
  unique_hosts: 1
current_memory: 314.80 KiB
total_memory_allocated: 14.85 MiB
...
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Using the sys Schema Example 1

3. Which statements are causing this activity?

```
mysql> SELECT * FROM user_summary_by_statement_type
-> WHERE user = 'bobsmith'\G
***** 1. row *****
    user: bobsmith
  statement: alter_table
      total: 262
total_latency: 28.87 s
  max_latency: 1.54 s
lock_latency: 22.56 s
  rows_sent: 0
rows_examined: 0
rows_affected: 640
  full_scans: 0
***** 2. row *****
    user: bobsmith
  statement: execute_sql
      total: 398
total_latency: 23.63 s
  max_latency: 339.61 ms
lock_latency: 0 ps
  rows_sent: 0
...
...
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Using the sys Schema: Example 2

Question: “Which statements are using temporary tables on disk?”

```
mysql> SELECT * FROM statements_with_temp_tables LIMIT 5\G;
***** 1. row *****
    query: SELECT `r` . `trx_wait_started ...
          db: NULL
    exec_count: 478707
  total_latency: 13.43 m
memory_tmp_tables: 3350949
disk_tmp_tables: 957414
avg_tmp_tables_per_query: 7
tmp_tables_to_disk_pct: 29
  first_seen: 2016-10-10 09:49:18
  last_seen: 2016-10-10 13:12:36
        digest: 529f34e4046a3f19b7023aca37d6a394
***** 2. row *****
    query: SELECT `t` . `THREAD_ID` AS `t ...
          db: NULL
    exec_count: 20835
  total_latency: 38.96 m
memory_tmp_tables: 83340
disk_tmp_tables: 41666
avg_tmp_tables_per_query: 4
...
...
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

- Monitoring MySQL with Log Files
- Monitoring MySQL with Status Variables
- Monitoring MySQL with Performance Schema
- **MySQL Enterprise Audit**
- MySQL Enterprise Monitor
- Monitoring User Activity



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL Enterprise Audit

- Implemented using the `audit_log` server plugin together with other components
 - Available with Enterprise Edition subscriptions
- Uses the new rule-based filtering to replace the legacy policy-based logging
 - Can assign a different filter to each user account
- Can be written into an XML or JSON formatted file
- Produces an audit record of the server activity in a log file
 - Contents depend on the filtering rules and can include:
 - A record of errors that occur on the system
 - When clients connect and disconnect
 - What actions they perform while connected
 - Which databases and tables they access



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Installing MySQL Enterprise Audit

- Run the SQL script found in the `share` directory of MySQL installation:
 - `audit_log_filter_win_install.sql`: for Windows system
 - `audit_log_filter_linux_install.sql`: for Linux/Unix-based systems
- It will install the following components:
 - A server-side plugin named `audit_log` examines auditable events and determines whether to write them to the audit log.
 - User-defined functions enable manipulation of filtering definitions that control logging behavior, the encryption password, and log file reading.
 - Tables in the `mysql` system database provide persistent storage of filter (`audit_log_filter`) and user account (`audit_log_user`) data. If these tables are missing, the `audit_log` plugin operates in the legacy policy-based logging.
 - System variables enable audit log configuration and status variables provide runtime operational information.
 - An `AUDIT_ADMIN` privilege enables users to administer the audit log.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Once installed, MySQL Enterprise Audit remains installed until uninstalled.

To remove it, execute the following statements:

```
DROP TABLE IF EXISTS mysql.audit_log_filter;
DROP TABLE IF EXISTS mysql.audit_log_user;
UNINSTALL PLUGIN audit_log;
DROP FUNCTION audit_log_filter_set_filter;
DROP FUNCTION audit_log_filter_remove_filter;
DROP FUNCTION audit_log_filter_set_user;
DROP FUNCTION audit_log_filter_remove_user;
DROP FUNCTION audit_log_filter_flush;
DROP FUNCTION audit_log_encryption_password_get;
DROP FUNCTION audit_log_encryption_password_set;
DROP FUNCTION audit_log_read;
DROP FUNCTION audit_log_read_bookmark;
```

Audit Log File Configuration

- The `audit_log` server option enables or disables the `audit_log` plugin at startup.
 - Set to `FORCE_PLUS_PERMANENT` to prevent the plugin from being removed while the server is running.
- The log file is named `audit.log` and, by default, is in the server data directory.
 - To change the name or location of the file, set the `audit_log_file` system variable at server startup.
- If you set the `audit_log_rotate_on_size` server option to a number greater than 0, the log file is rotated when the size is reached.
- To balance compliance with performance, use the `audit_log_strategy` server option to choose between:
 - `SYNCHRONOUS`
 - `SEMISYNCHRONOUS`
 - `ASYNCHRONOUS`
 - `PERFORMANCE`



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

For security reasons, write the audit log file to a directory accessible only to the MySQL server and to users with a legitimate reason to view the log. The audit log can be encrypted to protect against unauthorized access.

Audit Log Filtering

Steps to enable audit log filtering:

1. Define a filter

- `audit_log_filter_set_filter(filter_name, definition)`
 - The filter definition is specified using JSON document.
 - Stored in the `mysql.audit_log_filter` table

2. Assign a filter to a user account

- `audit_log_filter_set_user(user_name, filter_name)`
 - Specify the `user_name` as `user_name@host_name` format or `%` for any user account that has no explicitly assigned filter.
 - Only one filter can be assigned to a user account.
 - Stored in the `mysql.audit_log_user` table

The SUPER privilege is required to run these audit filtering functions.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

To remove a filter:

- `audit_log_filter_remove_filter(filter_name)`

To remove a user account filter assignment:

- `audit_log_filter_remove_user(user_name)`

By default, rule-based audit log filtering logs no auditable events for any users. To log all auditable events for all users, use the following statements:

```
SELECT audit_log_filter_set_filter('log_all',
                                    '{ "filter": { "log": true } }');
SELECT audit_log_filter_set_user('%', 'log_all');
```

Audit Log Filter Definitions

- Logging all events

```
{ "filter": { "log": true } }
```

- Logging a specific event class

```
{ "filter": { "class": { "name": "connection" } } }
```

- Logging multiple event classes

```
{ "filter": { "class": [ { "name": "connection" }, { "name": "table_access" } ] } }
```

- Logging specific event subclasses

```
{ "filter": { "class": { "name": "table_access", "event": [ { "name": "insert" }, { "name": "delete" }, { "name": "update" } ] } } }
```

- Blocking execution of specific events

```
{ "filter": { "class": { "name": "table_access", "event": { "name": [ "insert", "update", "delete" ], "abort": true } } } }
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Statements matched and blocked by the filter return an error to the client:

- ERROR 1045 (28000): Statement was aborted by an audit log filter

See <https://dev.mysql.com/doc/refman/en/audit-log-filter-definitions.html> for other constructs of the filter definitions.

Audit Log File Format

Is controlled by the `audit_log_format` system variable at server startup. The available formats are:

- NEW: New-style XML format that has better compatibility with Oracle Audit Vault
 - This is the default in MySQL 8.0.
- OLD: Old-style XML format that is the original format used in the first version of Enterprise Audit
- JSON: JSON format
 - Allows bookmarking and reading the audit log files using user-defined functions
 - `audit_log_read_bookmark()` returns a JSON string representing a bookmark for the most recently written audit log event.
 - `audit_log_read()` reads events from the audit log and returns a JSON string containing an array of audit events.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

An example of `audit_log_read()` invocation using the current bookmark:

```
mysql> SELECT audit_log_read(audit_log_read_bookmark());  
+-----+  
| audit_log_read(audit_log_read_bookmark()) |  
+-----+  
| [ {"timestamp":"2019-10-03 22:41:24","id":0,"class":"connection", ... |  
+-----+
```

Note: Audit log bookmarking and reading only apply to JSON formatted file. These functions return an error if the audit log format is not JSON.

Audit Log File: New-Style XML Format

- The root element is <AUDIT>.
- Each audit event is an <AUDIT_RECORD> element.
- XML elements are used to specify the event details.

```
<?xml version="1.0" encoding="utf-8"?>
<AUDIT>
  ...
  <AUDIT_RECORD>
    <TIMESTAMP>2019-10-03T14:09:38 UTC</TIMESTAMP>
    <RECORD_ID>6_2019-10-03T14:06:33</RECORD_ID>
    <NAME>Query</NAME>
    <CONNECTION_ID>5</CONNECTION_ID>
    <STATUS>0</STATUS>
    <STATUS_CODE>0</STATUS_CODE>
    <USER>root[root] @ localhost [127.0.0.1]</USER>
    <OS_LOGIN/>
    <HOST>localhost</HOST>
    <IP>127.0.0.1</IP>
    <COMMAND_CLASS>drop_table</COMMAND_CLASS>
    <SQLTEXT>DROP TABLE IF EXISTS t</SQLTEXT>
  </AUDIT_RECORD>
  ...
</AUDIT>
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The following is an example of the old-style XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<AUDIT>
  ...
  <AUDIT_RECORD
    TIMESTAMP="2019-10-03T14:25:24 UTC"
    RECORD_ID="6_2019-10-03T14:25:00"
    NAME="Query"
    CONNECTION_ID="4"
    STATUS="0"
    STATUS_CODE="0"
    USER="root[root] @ localhost [127.0.0.1]"
    OS_LOGIN=""
    HOST="localhost"
    IP="127.0.0.1"
    COMMAND_CLASS="drop_table"
    SQLTEXT="DROP TABLE IF EXISTS t"/>
  ...
</AUDIT>
```

Audit Log File: JSON Format

- The audit log file contents form a JSON array.
- Each array element represents an audited event as a JSON object of key-value pairs.

```
[  
  ...  
  { "timestamp": "2019-10-03 14:23:41",  
    "id": 0,  
    "class": "table_access",  
    "event": "insert",  
    "connection_id": 5,  
    "account": { "user": "root", "host": "localhost" },  
    "login": { "user": "root", "os": "", "ip": "127.0.0.1", "proxy": "" },  
    "table_access_data": { "db": "test",  
                           "table": "t1",  
                           "query": "INSERT INTO t1 (i) VALUES(1), (2), (3)",  
                           "sql_command": "insert" } },  
  ...  
]
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Audit Record Values

- The `TIMESTAMP` of each audit record is in UTC.
- The `NAME` value represents the type of event. Some common values:
 - “Connect” indicates a login event.
 - “Quit” indicates a client disconnect.
 - “Query” indicates an SQL statement is executed.
 - “Audit” and “NoAudit” indicate the points at which auditing starts and stops.
- The `STATUS` value provides the command status.
 - This is the same as the `Code` value displayed by the MySQL command `SHOW ERRORS`.
- Some values appear only for specific event types.
 - For example, a “Connect” event includes attributes such as `HOST`, `DB`, `IP`, and `USER`, and a “Query” event includes the `SQLTEXT` value.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

List of mandatory items:

- `NAME`
- `RECORD_ID`
- `TIMESTAMP`

List of optional items:

- | | |
|--------------------------------|--------------------------------|
| • <code>COMMAND_CLASS</code> | • <code>CONNECTION_ID</code> |
| • <code>CONNECTION_TYPE</code> | • <code>DB</code> |
| • <code>HOST</code> | • <code>IP</code> |
| • <code>MYSQL_VERSION</code> | • <code>OS_LOGIN</code> |
| • <code>OS_VERSION</code> | • <code>PRIV_USER</code> |
| • <code>PROXY_USER</code> | • <code>SERVER_ID</code> |
| • <code>SQLTEXT</code> | • <code>STARTUP_OPTIONS</code> |
| • <code>STATUS</code> | • <code>STATUS_CODE</code> |
| • <code>USER</code> | • <code>VERSION</code> |

Note: These are item names for XML formatted file. JSON formatted file has a different list of item names. See <https://dev.mysql.com/doc/refman/en/audit-log-file-formats.html> for the complete list.

Quiz



Which of the following my.cnf server options enables the audit log?

- a. audit-log=ON
- b. enable-audit_log
- c. log=AUDIT
- d. plugin-load=audit_log.so



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Answer: d

Topics

- Monitoring MySQL with Log Files
- Monitoring MySQL with Status Variables
- Monitoring MySQL with Performance Schema
- MySQL Enterprise Audit
- **MySQL Enterprise Monitor**
- Monitoring User Activity



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL Enterprise Monitor

MySQL Enterprise Monitor is a key component of MySQL Enterprise Edition.

- Connects to one or more MySQL servers
- Reads performance and configuration metrics from status variables and information tables
- Uses a MySQL database to store its repository
- Features include:
 - Continuous monitoring
 - Including replication and cloud instances
 - Automatic alerts
 - Advisors
 - Visual query analysis and graphs
 - Account management



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

MySQL Enterprise Monitor has many modules, and this lesson provides only a brief overview of some of the main ones.

Installing MySQL Enterprise Monitor

- Available commercially from Oracle Software Delivery Cloud or My Oracle Support
- Installed from binaries
- Comprises:
 - Agents:
 - Run on MySQL server hosts in the network
 - Communicate monitoring data to the Service Manager
 - Service Manager:
 - Runs the configurable web application
 - Monitors server agents, displays graphs, and transmits alerts
- Service Manager requires a MySQL server instance to manage its history and configuration. This can be:
 - An existing instance
 - A bundled instance (installed with Enterprise Monitor) that runs on its own port



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Installing the Service Manager

- Select the installation language.
 - English, Japanese, or Simplified Chinese
- Choose the installation directory.
 - Default: /opt/mysql/enterprise/monitor
- Choose initial configuration based on system size:
 - **Small system:** No more than 5–10 MySQL servers monitored from a laptop computer or a low-end server with no more than 4 GB of RAM
 - **Medium system:** Up to 100 MySQL servers monitored from a medium-size, shared server with 4–8 GB of RAM
 - **Large system:** More than 100 MySQL servers monitored from a high-end server dedicated to MySQL Enterprise Monitor with more than 8 GB RAM



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Installing the Service Manager

1. Choose the Tomcat ports (HTTP and SSL).
 - Defaults: 18080 and 18443, respectively
2. Choose the user account with which the Service Manager logs in to the operating system.
 - Created by the installer if it does not already exist
 - Default: mysqlmem
3. Choose the repository instance.
 - Create a new (bundled) MySQL server instance or use an existing MySQL server.
4. Provide connection information for the repository.
 - Username, password, port, database name, SSL
 - The installer creates and configures the instance with this information if you select the bundled server.
 - Server name if you select an existing server



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Bundled Software

The Service Manager bundles Tomcat as its web application server. You can choose different ports if you already have applications running on the default ports.

It also bundles an instance of MySQL server. This instance is dedicated for use by Enterprise Monitor and runs on a different port. If you choose not to install the bundled server, you must specify the location (host name and port or socket) of a compatible MySQL instance, along with the credentials of a user account on that instance that has permissions to create and access the repository.

The developers recommend that you choose the bundled instance, because it has a configuration designed to run optimally with Enterprise Monitor.

Post-Installation Configuration

After you install the Service Manager, connect to `https://hostname:18443/`.

- Use the name of the host on which you installed the Service Manager and the configured Tomcat SSL port.
- Ensure that the host's firewall permits connections on that port.
- The Service Manager uses a self-signed certificate by default.
 - Replace this with a certificate recognized by your organization's browsers.
 - If you do not have such a certificate, add a security exception to enable your browser to connect to the server.
- Configure the Manager and Agent accounts:
 - **Manager:** Uses the Enterprise Monitor interface
 - **Agent:** Connects from agents to the Service Manager



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Installing Agents

- Select the installation language.
 - English, Japanese, or Simplified Chinese
- Choose the installation directory.
 - Default: /opt/mysql/enterprise/agent
- Choose how the agent connects to the MySQL instance that it monitors.
 - TCP/IP or Socket
 - Specify connections during installation or from the Service Manager during use.
 - If you choose to specify connections during installation, you must provide credentials and server coordinates.
- Specify the IP address of the Service Manager.
 - Provide the credentials of the agent user.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The Service Manager contains a bundled agent that monitors MySQL servers on the same host. You do not need to install a separate agent on that host.

MySQL Enterprise Monitor: Managing Multiple Servers

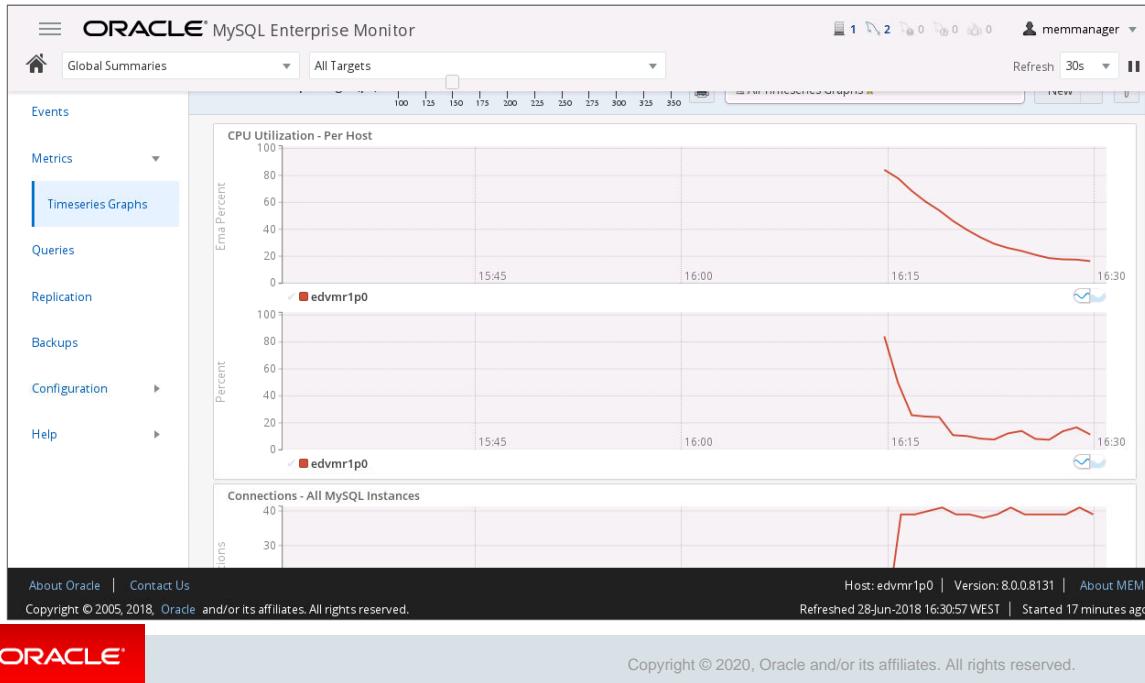
The screenshot shows the MySQL Instances dashboard in MySQL Enterprise Monitor. The interface includes a left sidebar with navigation links like Overview, Events, Metrics, Queries, Replication, Backups, Configuration, Instances (which is selected), Groups, and Advisors. The main area displays a table of MySQL instances. A red box highlights the top-right corner of the screen, labeled 'Indicator area', which contains status icons for monitoring and connectivity. Another red box highlights the 'Filter instance drop-down' button above the instance table. The table has columns for Instance, Notes, Versions, Agent, Operating System, Port, and Data Dir. It lists two monitored instances: 'edvmr1p0:3306' and 'edvmr1p0:13306', and two unmonitored instances: 'edvmr1p0:3306' and 'edvmr1p0:13306'. The bottom of the screen shows copyright information and an Oracle logo.

Instance	Notes	Versions	Agent	Operating System	Port	Data Dir
All (2/2)						
edvmr1p0:3306	8.0.11-commercial	8.0.0.8131	Oracle Linux 7.3	3306	/var/lib/mysql/	
edvmr1p0:13306	5.7.22-enterprise-co...	8.0.0.8131	Oracle Linux 7.3	13306	/opt/mysql/enterpri...	
Ungrouped (2/2)						
edvmr1p0:3306	8.0.11-commercial	8.0.0.8131	Oracle Linux 7.3	3306	/var/lib/mysql/	
edvmr1p0:13306	5.7.22-enterprise-co...	8.0.0.8131	Oracle Linux 7.3	13306	/opt/mysql/enterpri...	

The screenshot in the slide shows the MySQL Instances dashboard. The list of monitored instances displays by default. If there are unmonitored instances, bad connections, or unreachable agents, then MySQL Enterprise Monitor alerts you in the indicator area in the top-right corner of the page. You can view details of these instances by clicking one of the indicators or selecting the appropriate entry in the instances filter drop-down list shown in the slide. A new section appears on the page that lists those instances.

If you have unmonitored instances, then you can monitor them by providing the connection details and credentials of an account with privileges to monitor the instances, or you can explicitly ignore the instances.

MySQL Enterprise Monitor: Timeseries Graphs



The Timeseries Graphs page displays a range of graphs that provide a visual display of MySQL metrics.

MySQL Enterprise Monitor: Advisors

The screenshot shows the MySQL Enterprise Monitor interface. The left sidebar has a tree view with nodes like Global Summaries, All Targets, Overview, Events, Metrics, Queries, Replication, Backups, Configuration, Instances, Groups, and Advisors (which is selected). The main panel is titled "Manage Advisors" and shows a table of rules under the "Memory Usage" category. The table has columns for Item, Info, Coverage, Schedule, Event Handling, and Parameters. Three specific rows are highlighted with orange circles and numbers: 1 points to the "Memory Usage" category header; 2 points to the first row ("InnoDB Buffer Cache Has Sub-Optimal Hit Rate"); and 3 points to the last row ("Query Cache Potentially Undersized"). The Parameters column for row 2 shows thresholds: 95 (Notice), 85 (Warning), and 75 (Critical). The Parameters column for row 3 shows thresholds: 60 (Notice), 50 (Warning), and 40 (Critical).

Item	Info	Coverage	Schedule	Event Handling	Parameters
InnoDB Buffer Cache Has Sub-Optimal Hit Rate	100% (2/2)	5m	0 0 0	0 0 0	95, 85, 75
Key Buffer Size May Not Be Optimal For Key Cache	100% (2/2)	5m	0 0 0	0 0 0	95, 85, 75
Query Cache Has Sub-Optimal Hit Rate	100% (2/2)	5m	0 0 0	0 0 0	60, 50, 40
Query Cache Potentially Undersized	100% (2/2)	5m	0 0 0	0 0 0	0

MySQL Enterprise Monitor contains many advisors that enable you to monitor various aspects of running MySQL servers in your enterprise. The screenshot shows the list of advisors organized by category and rule. You can configure these advisors by using the Configuration > Advisors menu item.

1. Each advisor consists of a set of rules in a category defined by the advisor. For example, the Memory Usage category contains six rules.
2. Each rule consists of an expression made up of attributes available to MySQL Enterprise Monitor agents, along with configured parameters and a schedule on which the rule runs. For example, the “InnoDB Buffer Cache Has Sub-Optimal Hit Rate” rule calculates the ratio between the `Innodb_buffer_pool_reads` and `Innodb_buffer_pool_read_requests` status variables. This rule runs every five minutes by default.
3. Each rule has parameters that define thresholds for rule expressions. In the case of the “InnoDB Buffer Cache Has Sub-Optimal Hit Rate” rule, if its expression returns a value less than 95, that triggers the “Notice” threshold. A value less than 85 triggers the “Warning” threshold, and a value less than 75 triggers the “Critical” threshold. You can configure “Emergency” thresholds for rules that require an extra level of response.

MySQL Enterprise Monitor: Events

The screenshot shows the MySQL Enterprise Monitor interface. The top navigation bar includes the Oracle logo, the title "MySQL Enterprise Monitor", and user information "memmanager". The left sidebar has a tree view with nodes like Global Summaries, All Targets, Overview, Events, Metrics, Queries, Replication, Backups, Configuration, Instances, Groups, Advisors, Email Notification Groups, Email Settings, and Email Notification Status. The main content area is titled "Event Handlers" and contains a table with one entry: "Handler Name: Default Auto-clo..., State: Active, Subjects: All, Advisors: All, Statuses: All, Actions: Auto Close (honor advi...)".

Advisors raise events when a condition is met. The Event Handlers page allows you to respond to specific events by creating a handler for it. Creating a handler involves specifying the condition that should cause the event to fire and details about how you would like to be notified. The Event Handlers can be configured to send email and SNMP notifications.

Topics

- Monitoring MySQL with Log Files
- Monitoring MySQL with Status Variables
- Monitoring MySQL with Performance Schema
- MySQL Enterprise Audit
- MySQL Enterprise Monitor
- Monitoring User Activity



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

SHOW PROCESSLIST

- **SHOW PROCESSLIST** shows you which process threads are running on connections.
 - The **PROCESS** privilege permits you to see threads for all connections, not just your own.
- **SHOW PROCESSLIST** produces the following columns:
 - **Id**: Connection identifier
 - **User**: MySQL user who issued the statement
 - **Host**: Host name of the client issuing the statement
 - **db**: Default database selected, otherwise **NULL**
 - **Command**: Type of command that the thread is executing
 - **Time**: Seconds that the thread has been in its current state
 - **State**: Action, event, or state indicating what the thread is doing
 - **Info**: First 100 characters of the associated statement or **NULL**
 - Use **SHOW FULL PROCESSLIST** to see the full statement.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

You can also get thread information from the `INFORMATION_SCHEMA.PROCESSLIST` table or the `mysqladmin processlist` command. If you do not have the `PROCESS` privilege, you can view only your own threads. That is, you can view only those threads associated with the MySQL account that you are using.

A mutex locks the connections to retrieve the required information. This can affect the performance.

Performance Schema Threads Table

- Contains a row for each server thread
- Has all the information returns by `SHOW PROCESSLIST` and more
 - Columns prefixed with `PROCESSLIST_` provide the same information as `SHOW PROCESSLIST`.
- Includes both the user connections and background threads
- Indexed to speed up table scanning
- Does not require a mutex and has minimal impact on server performance
 - Better alternative to `SHOW PROCESSLIST`
- Queried using a `SELECT` statement:

```
SELECT * FROM performance_schema.threads
```

- Can use other clauses like `WHERE`, `GROUP BY`, `ORDER BY`, and `LIMIT`



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The `sys` schema has the `processlist` and `x$processlist` views which provide the information similar to the `threads` table with additional status information of the current and last statements executed by the threads.

Killing Processes

- The Ctrl + C key combination terminates a running statement in the mysql command-line client.
- Use the **KILL *id*** statement to kill processes.

```
mysql> KILL 31;  
Query OK, 0 rows affected (#.## sec)
```

- Use **mysqladmin kill *id*** to kill the thread with the specified ID.

```
# mysqladmin -uroot -p kill 31  
Enter password: password
```

- Killing a process terminates the statement and the client connection.
 - The client must reconnect to issue its next statement.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

If you specify a nonexistent thread ID when you execute **KILL** or **mysqladmin kill**, you get an “Unknown thread id” error. This error also occurs if you try to kill the same thread twice after the first attempt succeeds.

Limiting User Activity

- Limit the use of server resources by setting the global `max_user_connections` variable to a nonzero value.
 - This limits the number of simultaneous connections by any one account, but does not limit what a client can do when connected.
- Limit the following server resources for individual accounts:
 - `max_queries_per_hour`: The number of queries that an account can issue per hour
 - `max_updates_per_hour`: The number of updates that an account can issue per hour
 - `max_connections_per_hour`: The number of times an account can connect to the server per hour
 - `max_user_connections`: The number of simultaneous connections allowed



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Setting Resource Limits

- To set resource limits for an account, use an `ALTER USER` statement containing a `WITH` clause that names each resource to be limited.
 - The default value for each limit is zero, indicating no limit.
- Provide resource limits in the `WITH` clause in any order
- For example, to set limits for the local user Bob to access the customer database, issue the following statement:

```
mysql> ALTER USER 'bob'@'localhost'  
      ->     WITH MAX_QUERIES_PER_HOUR 20  
      ->           MAX_UPDATES_PER_HOUR 10  
      ->           MAX_CONNECTIONS_PER_HOUR 5  
      ->           MAX_USER_CONNECTIONS 2;
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Resetting Limits to Default Values

- Set the `max_user_connections` limit to 0 to set it to the global default.
 - This indicates that the maximum number of simultaneous connections allowed for the specified account is the global value of the `max_user_connections` system variable.
- To reset an existing limit for any of the per-hour resources to the default of “no limit,” specify a value of 0, as in the following example:

```
mysql> ALTER USER 'erica'@'localhost'  
-> WITH MAX_CONNECTIONS_PER_HOUR 0;
```

- This statement does not affect any other per-account limits on the specified account.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Summary



In this lesson, you should have learned how to:

- Configure and view MySQL log files
- Identify slow queries
- Configure MySQL Enterprise Audit
- Use MySQL variables to monitor activity in MySQL
- Use MySQL Enterprise Monitor to view activity in MySQL



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Practices

- 5-1: Configuring the Slow Query Log
- 5-2: Using Performance Schema
- 5-3: Installing MySQL Enterprise Monitor
- 5-4: Monitoring Server Activity



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

6

Managing MySQL Users



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Objectives



After completing this lesson, you should be able to:

- Create user accounts and roles
- Design a permissions structure
- Control user and role permissions
- Grant access to system operations
- Use authentication plugins
- Expire accounts manually and automatically



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

- MySQL Privilege System
 - Creating and Modifying User Accounts and Roles
 - Configuring Passwords and Account Expiration
 - Authentication Plugins
 - Granting Permissions
 - Activating Roles
 - Grant Tables



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Importance of User Management

Managing users in MySQL gives you the ability to control what the users can or cannot do.

- Create user accounts and roles with different privileges that are appropriate to their function.
- Avoid using the `root` account to:
 - Constrain compromised applications
 - Protect against mistakes during routine maintenance
- Ensure data integrity by proper assignment of individual user privileges
 - Permit authorized users to do their work
 - Prevent unauthorized users from accessing data beyond their privileges



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Authentication and Authorization

When you connect to a MySQL server and execute a query, it authenticates you and authorizes your activity.

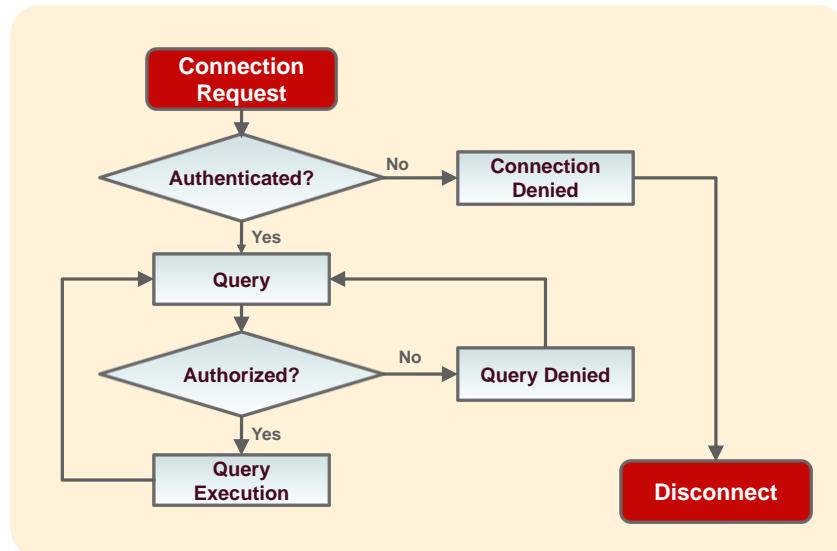
- **Authentication:** Verifies the user's *identity*
 - This is the first stage of access control.
 - You must successfully authenticate each time you connect.
 - If you fail to authenticate, your connection fails, and your client disconnects.
- **Authorization:** Verifies the user's *privileges*
 - This second stage of access control takes place for each request on an authenticated connection.
 - MySQL determines:
 - What operation you want to perform
 - Whether you have sufficient privileges to perform that operation



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

User Connection and Query Process



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Viewing User Account Settings

- Query the mysql database to view user identification information:

```
mysql> SELECT user, host, authentication_string FROM mysql.user\G
***** 1. row *****
    user: root
    host: localhost
authentication_string: $A$005$j<(4`"B]1b{Nv;P{RWHx/WNwB.nEfslHSyye3xB8t6FaYwm2V./
zZI76eYw2
...
```

- Each row identifies a single account.
- Fields that identify accounts include:
 - User: The user's name on this account
 - Host: DNS host name or IP address from which that user can connect
 - Authentication_string: The password that the user must provide for this account, encrypted according to the scheme implemented by the account's authentication plugin



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Pluggable Authentication

- In MySQL 8.0, `caching_sha2_password` is the default authentication plugin.
- During connection using the `caching_sha2_password` plugin, MySQL uses the following to authenticate an account:
 - Username
 - Password
 - Client host
- When specifying host names, remember the proper perspective:
 - Specify the server's host name when connecting using a client.
 - Specify the client's host name when adding a user to the server.



ORACLE®

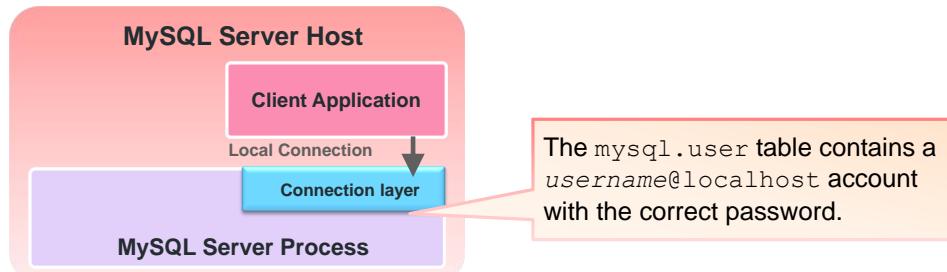
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Local Connection

- To connect to the local server by using the `mysql` client, specify the username and password for the account that you want to use:

```
mysql -u username -ppassword
```

- The default host name is `localhost`, which indicates a Unix socket connection on Unix-like systems.
- Specify the `--protocol=TCP` or `-h 127.0.0.1` option to connect using TCP/IP.



ORACLE

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

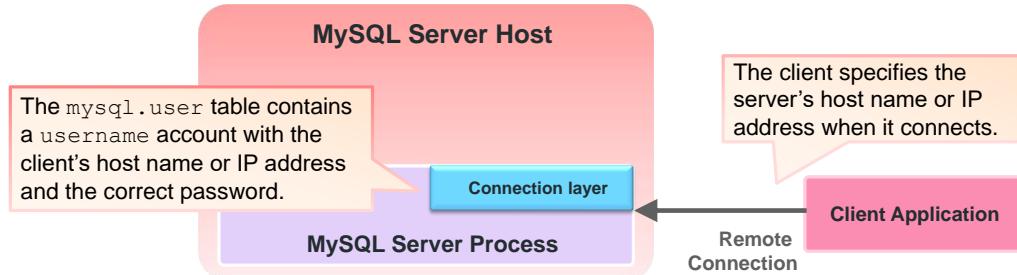
There is no space between the `-p` option and the password. If there is a space after `-p` option, it will prompt for the password instead of reading the password from the command line.

Remote Connection

- To connect to a server that is not installed on your client's local host, provide the host name of the server to which you are connecting:

```
mysql -u username -ppassword -h servername
```

- The host name associated with your user in the `mysql.user` table refers to the name of the client host from which you are connecting, not the server host.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Topics

- MySQL Privilege System
- **Creating and Modifying User Accounts and Roles**
- Configuring Passwords and Account Expiration
- Authentication Plugins
- Granting Permissions
- Activating Roles
- Grant Tables



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Account Names

- Consist of:
 - Username (which need not be unique)
 - Name or IP address of the client host from which the user connects to the server
- Have the format '`username'@'hostname'`
 - Usernames can be up to 32 characters long.
 - If a username or host name is valid as an unquoted identifier, the quotation marks are optional.
 - You must use single quotation marks around usernames and host names if they contain special characters, such as dashes.
 - Examples:
 - `root@localhost` ('`root`'@'`localhost`' is also valid.)
 - `reports_app@'server-1'` ('`reports_app`'@'`server-1`' is also valid.)
 - `'%^&'@192.168.5.3` ('%^&'@'`192.168.5.3`' is also valid.)



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Note

The conditions for valid unquoted identifiers are described at
<https://dev.mysql.com/doc/mysql/en/identifiers.html>.

Host Name Patterns

Examples of permissible host name formats:

- Host name: `server1`
- Qualified host name: '`server2.example.com`'
- IP address: `192.168.9.78`
- IP network: '`10.0.0.0/255.255.255.0`'
- Wildcards: `%` (multicharacter) or `_` (single character)
 - In IP address: '`192.168.%`'
 - In qualified host name: '`%.example.com`'
 - Single character wildcard: '`server_.example.com`'
 - If a host matches two or more patterns, MySQL chooses the most specific pattern.
 - An account with a host name '`%`' can connect from any host.
 - Default if you do not specify a host name when you create a user.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Wildcards do not apply to usernames. For example, the user account `reports_app@'server-1'` does not enable connections with usernames such as `reportsXapp` or `reportsYapp`.

Creating a User Account

- Provide a user and host for each user account.
- For example, use the **CREATE USER... IDENTIFIED BY** statement to create an account:
 - For a user named `webuser`
 - To connect from `localhost`
 - Using the password `Abc123`

```
CREATE USER webuser@localhost IDENTIFIED BY 'Abc123';
```

- Avoid possible security risks when creating accounts:
 - Do not create accounts without a password.
 - Do not create anonymous accounts.
 - Accounts with no username such as ''@`localhost`
 - Where possible, avoid wildcards when you specify account host names.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Avoid using wildcards in host names except where it is strictly necessary and properly audited to avoid abuse or accidental exposure. Run periodic checks as follows:

```
mysql> SELECT User, Host FROM mysql.user WHERE Host LIKE '%\%%';
```

Roles

Roles are:

- Collections of privileges
 - Can be granted to user accounts and other roles like individual privileges
 - Can also contain other roles
- A more convenient method of adding, removing, and managing grants
- Similar to users
 - They are stored in the `mysql.users` table.
 - Role names consist of user and host (`'rolename'@'hostname'`).
 - The difference is you cannot log in as a role.
 - User accounts can act as roles and be granted to other users.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Roles are implemented in a similar way to users, except you cannot log in as a role. You even provide a name and host when creating a role, in the same way as you would when creating a user. However, unlike users, `rolename` cannot be blank, and unlike users, the `%` character in `hostname` does not have any wildcard properties.

Creating a Role

- Use the **CREATE ROLE** statement to create one or more roles.
 - This example creates two roles:
 - r_admin
 - r_dev@localhost
- ```
CREATE ROLE r_admin, r_dev@localhost;
```
- The hostname defaults to '%' if omitted.
  - A role is created as an account that
    - Is locked
    - Has no password
    - Is assigned the default authentication plugin
    - Is stored in the mysql.user table



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Role names and account names must be distinct.

It is recommended to prefix the role names with 'r\_' to differentiate role names from user account names.

A role is an account that does not allow connections because it is created as a locked account. You can use the **ALTER USER** statement to unlock a role, converting it into an account that allows connection.

```
ALTER USER r_admin IDENTIFIED BY 'password' ACCOUNT UNLOCK;
```

## Manipulating User Accounts and Roles

- Use the **RENAME USER** statement to rename user accounts and roles:

```
RENAME USER consultant@laptop3 TO james@laptop3 , r_admin TO r_super;
```

- Changes an existing account name or role name
- Changes either the username or host name parts or both

- Use the **DROP USER** or **DROP ROLE** statement to remove user accounts and roles:

```
DROP USER james@laptop3;
```

```
DROP ROLE r_super, r_dev@localhost;
```

- Revokes all privileges for an existing account or role
- Revokes the role from any accounts or roles to which the role was granted
- Removes the account or role
- Deletes all records for the account from any grant table in which they exist

- Rename or remove users and roles when their access requirements change.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

You can also use **DROP USER** to remove roles and **DROP ROLE** to remove users.

## Topics

- MySQL Privilege System
- Creating and Modifying User Accounts and Roles
- **Configuring Passwords and Account Expiration**
- Authentication Plugins
- Granting Permissions
- Activating Roles
- Grant Tables



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Setting the Account Password

- When you create a user:
  - CREATE USER ... IDENTIFIED BY '*newpassword*'
- For an existing user:
  - ALTER USER ... IDENTIFIED BY '*newpassword*' or  
SET PASSWORD [FOR *user*] = '*newpassword*'
    - Specify the username explicitly:

```
ALTER USER webuser1@localhost IDENTIFIED BY 'n3W%P4$$w0rd';
SET PASSWORD FOR webuser2@localhost = 'n3W%P4$$w0rd';
```
    - Change the password for the current user:

```
ALTER USER USER() IDENTIFIED BY 'n3W%P4$$w0rd';
SET PASSWORD = 'n3W%P4$$w0rd';
```
  - mysqladmin ... password '*newpassword*'
    - Change the password for the user account you use to connect to the server:

```
mysqladmin -u webuser3 -p password 'n3W%P4$$w0rd';
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Dual Password Support

- Is available as of MySQL 8.0.14
- Allows a user account to have two passwords, designated as primary and secondary passwords
- Is enabled when setting a new password with the RETAIN CURRENT PASSWORD clause

```
ALTER USER USER() IDENTIFIED BY 'n3W%P4$$w0rd' RETAIN CURRENT PASSWORD;
```

- The new password becomes primary and existing password becomes secondary.

- Is removed with the statement:

```
ALTER USER USER() DISCARD OLD PASSWORD;
```

- The secondary password is removed.

- Is useful to minimize disruption when password is changed in scenarios like:
  - A system has a large number of MySQL servers, possibly involving replication
  - Multiple applications connect to different MySQL servers
  - Update the applications to use the new password



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Expiring Passwords Manually

- Create an account with an expired password with `CREATE USER ... PASSWORD EXPIRE`:

```
CREATE USER 'erika'@'localhost' IDENTIFIED BY 'first#Pass'
PASSWORD EXPIRE;
```

- The new user must change his or her password before executing any other statements.

- Expire a user's password with `ALTER USER ... PASSWORD EXPIRE`:

```
ALTER USER 'erika'@'localhost' PASSWORD EXPIRE;
```

- Users can connect to the MySQL server with expired passwords.
- Users must change their password before they can execute any other statements.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Configuring Password Expiration

- The `default_password_lifetime` global variable specifies the number of days after which passwords must be changed.
  - The default value is 0, which indicates that passwords do not expire.
- Set per-account password lifetime with the `PASSWORD EXPIRE` clause of `CREATE USER` or `ALTER USER`:

```
CREATE USER 'consultant'@'laptop3' IDENTIFIED BY 'change%me'
PASSWORD EXPIRE INTERVAL 30 DAY;
```

- Apply the default password lifetime to an account:

```
ALTER USER 'consultant'@'laptop3' PASSWORD EXPIRE DEFAULT;
```

- Disable automatic password expiration on an account:

```
ALTER USER 'consultant'@'laptop3' PASSWORD EXPIRE NEVER;
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Changing Expired Passwords

- If your password expires, you must change your password before you can execute any other statements.
  - All other statements that you execute return an error until you change your password, as in the following example:

```
mysql> SELECT * FROM City WHERE 1=2;
ERROR 1820 (HY000): You must SET PASSWORD before executing this statement
mysql> ALTER USER USER() IDENTIFIED BY 'newpass!!!';
Query OK, 0 rows affected (0.01 sec)
mysql> SELECT * FROM City WHERE 1=2;
Empty set (0.00 sec)
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Quiz



A user is logged in to his computer `david-laptop` as the Linux user `dave`. He logs in to MySQL by typing the following command:

```
mysql -u david -h dbserver2 -p
```

Assuming that the server uses `caching_sha2_password`, which of the following `mysql.user` accounts authenticates him?

- a. `dave@david-laptop`
- b. `dave@dbserver2`
- c. `david@david-laptop`
- d. `david@dbserver2`



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Topics

- MySQL Privilege System
- Creating and Modifying User Accounts and Roles
- Configuring Passwords and Account Expiration
- **Authentication Plugins**
- Granting Permissions
- Activating Roles
- Grant Tables



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Pluggable Authentication

MySQL supports a number of authentication mechanisms that are available through pluggable authentication.

- Plugins are built in or available as external libraries.
- Default server-side plugins are built in, always available, and include:
  - mysql\_native\_password:
    - Based on the password hashing method in use from before the introduction of pluggable authentication
  - sha256\_password:
    - Implements basic SHA-256 authentication
    - Is deprecated and will be removed in a future MySQL version
  - caching\_sha2\_password:
    - Implements SHA-256 authentication (like sha256\_password), but uses caching on the server side for better performance
    - Default authentication plugin in MySQL 8.0



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

You can set the `default_authentication_plugin` variable to change the default value.

## Cleartext Client-Side Authentication Plugin

The MySQL client library includes a built-in Cleartext Authentication plugin, `mysql_clear_password`. The plugin is:

- Used to send a plain text password to the server
  - It prevents the client from hashing the password.
- Enabled by:
  - The `LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN` environment variable
  - Specifying `--enable-cleartext-plugin` when running MySQL client applications such as `mysql` and `mysqladmin`
  - The `MYSQL_ENABLE_CLEARTEXT_PLUGIN` option of the `mysql_options()` C API function



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Some authentication methods, such as PAM (Pluggable Authentication Modules) authentication and LDAP authentication, require the client to send a plain text password to the server so that the server can process the password in its normal form. The `mysql_clear_password` plugin enables this behavior.

## Loadable Authentication Plugins

- Test Authentication plugin (**test\_plugin\_server**): Implements native and old password authentication
  - This plugin uses the `auth_test_plugin.so` file and is intended for testing and development purposes.
- Socket Peer-Credential (**auth\_socket**): Allows only MySQL users who are logged in via a UNIX socket from a UNIX account with the same name
  - This plugin uses the `auth_socket.so` file.
- No-login Authentication plugin (**mysql\_no\_login**): Prevents all client connections to any account that uses it
  - This plugin uses the `mysql_no_login.so` file.
  - Can be used for:
    - Accounts that must be able to execute stored programs and views without exposing those privileges to ordinary users
    - Proxied accounts that should never permit direct login



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

To load one of these plugins, start the server with the `plugin-load` option set to the plugin's file name. For example:

```
[mysqld]
plugin-load=auth_test_plugin.so
```

You can develop your own authentication plugins. The Test Authentication plugin is intended for use by developers to create their own plugins; its source code is available as part of the MySQL source code distribution.

## Enterprise Authentication Plugins

These authentication plugins are available in the MySQL Enterprise Edition commercial product. They support external authentication and proxy users.

- PAM Pluggable Authentication (`authentication_pam`):
  - Enables a standard interface to access various kinds of authentication methods, such as traditional Unix passwords or an LDAP directory
  - Works on Linux and macOS
- Windows Pluggable Authentication (`authentication_windows`):
  - Enables MySQL Server to use native Windows services to authenticate client connections
- LDAP Pluggable Authentication (`authentication_ldap_sasl`, `authentication_ldap_simple`):
  - Authenticates MySQL users by accessing directory services such as X.500
  - MySQL uses LDAP to fetch user, credential, and group information.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## PAM Authentication Plugin

- The PAM Authentication plugin is an Enterprise Edition plugin that authenticates MySQL accounts against the operating system.
- PAM defines services that configure authentication.
  - These are stored in /etc/pam.d.
- The plugin authenticates against:
  - Operating system users and groups
  - External authentication such as LDAP



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Configuring the PAM Authentication Plugin

- PAM looks in `/etc/pam.d` for services that it authenticates.
  - For example, to create a PAM service called `mysql-pam`, create the `/etc/pam.d/mysql-pam` file with the following content:

```
#%PAM-1.0
auth include password-auth
account include password-auth
```

- The preceding configuration performs simple Linux password authentication to PAM clients (including MySQL) that request the `mysql-pam` service.
- In addition to simple password authentication, PAM integrates with other authentication methods including LDAP and Active Directory, so you can use PAM to authenticate many services (including MySQL) against a single store in your network.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Creating Users that Authenticate with PAM

To create a MySQL user that maps directly to an operating system user, use a statement such as the following:

```
CREATE USER bob@localhost
IDENTIFIED WITH authentication_pam AS 'mysql-pam';
```

- This statement creates an account with:
  - User: bob
  - Host: localhost
  - Plugin: authentication\_pam
  - Authentication string: mysql-pam
- The plugin processes the authentication string, which applies the PAM rules contained in the mysql-pam PAM service.
- The password is not stored in MySQL's `user` table.
  - The operating system authenticates the password.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Creating PAM Proxied Users

The authentication string can include proxied mappings.

- Create an anonymous proxy user that uses PAM and maps from OS group to MySQL users.

```
CREATE USER ''@''
IDENTIFIED WITH authentication_pam
AS 'mysql-pam, www=webuser, root=root';
```

- In the preceding example:
  - Members of the operating system's `www` group are mapped to the MySQL `webuser` account.
  - Members of the `root` group are mapped to the MySQL `root` account.
- The proxy user must have the `PROXY` privilege on the mapped accounts:

```
GRANT PROXY ON webuser@localhost to ''@'';
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Logging In with PAM Accounts

- MySQL passes the username and password that it receives from the client to PAM, which authenticates against the operating system.
  - PAM can process the password only if it is in plain text. Enable the Cleartext client-side Authentication plugin.

```
mysql --enable-cleartext-plugin -u bob -p
Enter password: bob's OS password
```

- Proxied users assume the identity of the mapped account.
  - Example: Anne is not a MySQL user but is in the operating system's `www` group.

```
mysql --enable-cleartext-plugin -u anne -p
Enter password: anne's OS password
```

- Anne's client is now logged in with the privileges of the `webuser@localhost` account.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Users are still individually accountable when they use proxy accounts. MySQL tracks the logged-in user with the `USER()` function and the mapped user with the `CURRENT_USER()` function.

Example:

```
mysql> SELECT USER(), CURRENT_USER();
+-----+-----+
| USER() | CURRENT_USER() |
+-----+-----+
| anne@localhost | webuser@localhost |
+-----+-----+
1 row in set (0.00 sec)
```

Note that the `mysql.user` table does not contain an account for `anne@localhost`.

## Topics

- MySQL Privilege System
- Creating and Modifying User Accounts and Roles
- Configuring Passwords and Account Expiration
- Authentication Plugins
- **Granting Permissions**
- Activating Roles
- Grant Tables



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Authorization

The primary function of the MySQL authorization system is to associate an authenticated user with privileges on a database.

- After a user authenticates, MySQL asks the following questions to verify account privileges:
  - Who is the current user?
  - What privileges does that user have?
    - Administrative examples: SHUTDOWN, REPLICATION SLAVE, and LOAD DATA INFILE
    - Data examples: SELECT, INSERT, UPDATE, and DELETE
  - Where do these privileges apply?
    - Global, database, table, column, stored routine
- You must set up proper accounts and privileges for authorization to work.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Determining Appropriate User Privileges

- Grant privileges at the correct level of access:
  - Global
  - Database
  - Table
  - Column
  - Stored routine
- Grant privileges to users and roles according to their access requirements:
  - Read-only users: Global-, database-, or table-level privileges such as SELECT
  - Users who modify databases: Global-, database-, or table-level privileges such as INSERT, UPDATE, DELETE, CREATE, ALTER, and DROP
  - Administrative users: Global-level privileges such as FILE, PROCESS, SHUTDOWN, and SUPER



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Privilege Scope

- Grant several types of privileges to a MySQL account at different levels:
  - Globally or for particular databases, tables, or columns
  - Example: Give a user the ability to select from any table in any database by granting the user the **SELECT** privilege at the global level.
- Give an account complete control over a specific database without having any permissions on other databases.
  - The account can:
    - Create the database
    - Create tables and other database objects
    - Select from the tables
    - Add, delete, or update new records



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Granting Administrative Privileges

Administrative privileges are granted on the global level and enable the following activities:

- FILE: Run SQL statements that read and write files in the server host filesystem.
- PROCESS: Use the SHOW PROCESSLIST statement to see all statements that clients are executing.
- SHOW DATABASES: Use SHOW DATABASES statement to list all databases
- SHUTDOWN: Use the SHUTDOWN and RESTART statements, the mysqladmin shutdown command, and the mysql\_shutdown() C API function.
- RELOAD: Execute FLUSH statements to reload logs and privilege tables.

Grant administrative privileges sparingly, because they can be abused by malicious or careless users.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Other administrative privileges include CREATE ROLE, CREATE TABLESPACE, CREATE USER, DROP ROLE, REPLICATION CLIENT, REPLICATION SLAVE, and SUPER. The SUPER administrative privilege enables users to perform server-level tasks, including setting global variables, controlling logs and replication, and killing client connections.

Administrative privileges, including those in the slide, can compromise security, access privileged data, or perform denial-of-service attacks on a server. Ensure that you grant these privileges only to appropriate accounts.

## Dynamic Privileges

Dynamic privileges are runtime privileges defined during server startup or by a server component or plugin. Some of the dynamic privileges include:

- AUDIT\_ADMIN: Configure audit log settings in the `audit_log` plugin.
- FIREWALL\_ADMIN: Administer firewall rules in the `MYSQL_FIREWALL` plugin.
- GROUP\_REPLICATION\_ADMIN: Configure, start, and stop Group Replication.
- ROLE\_ADMIN: Grant and revoke roles and set the value of the `mandatory_roles` system variable.
- REPLICATION\_SLAVE\_ADMIN: Configure slave servers and start and stop replication.
- SYSTEM\_VARIABLES\_ADMIN: Change global system variables using `SET GLOBAL` and `SET PERSIST` statements.

Grant dynamic privileges with limited scope instead of `SUPER` privilege to implement principle of least privilege.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Other dynamic privileges include:

- APPLICATION\_PASSWORD\_ADMIN
- BACKUP\_ADMIN
- BINLOG\_ADMIN
- BINLOG\_ENCRYPTION\_ADMIN
- CONNECTION\_ADMIN
- ENCRYPTION\_KEY\_ADMIN
- FIREWALL\_USER
- PERSIST\_RO\_VARIABLES\_ADMIN
- RESOURCE\_GROUP\_ADMIN
- RESOURCE\_GROUP\_USER
- SERVICE\_CONNECTION\_ADMIN
- SESSION\_VARIABLES\_ADMIN
- SET\_USER\_ID
- TABLE\_ENCRYPTION\_ADMIN
- VERSION\_TOKEN\_ADMIN
- XA\_RECOVER\_ADMIN

Dynamic privileges are stored in the `mysql.global_grants` table.

## Special Privileges

- The `ALL` and `ALL PRIVILEGES` specifiers grant all privileges at the specified access level except the ability to give privileges to other accounts.
  - Use `GRANT ALL ON *.* ... WITH GRANT OPTION` to grant all privileges on the global level, including the ability to give privileges to other accounts.
- The `USAGE` specifier grants the ability to connect to the server.
  - This is the default privilege level for new accounts.
  - The account can access the server for limited purposes, such as issuing `SHOW VARIABLES` or `SHOW STATUS` statements.
  - The account cannot be used to access database contents such as tables; such privileges can be granted at a later time.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## GRANT Statement

- The GRANT statement assigns privileges or roles to MySQL user accounts and roles.

- Example GRANT syntax:

```
GRANT SELECT ON world.* TO r_viewer;
GRANT UPDATE, DELETE ON world.city TO kari@localhost;
```

- Statement clauses:

- Privileges to be granted
    - Example: SELECT, UPDATE, DELETE

- Privilege level:

- Global: \*.\*
    - Database: db\_name.\*
    - Table: db\_name.table\_name
    - Stored routine: db\_name.routine\_name

- Account or role to which you are granting the privilege



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

**Privileges available on the database level:** SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE TEMPORARY TABLE, LOCK TABLES, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE, EVENT, and TRIGGER.

**Privileges available on the table level:** SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE VIEW, SHOW VIEW, and TRIGGER.

## Granting Permissions on Columns

- Specify the column permission by listing columns after the permission is granted:

```
GRANT SELECT(user,host) ON mysql.user TO kari@localhost;
```

- The user or role has permission to perform the specified operation only on the columns listed.
- In the preceding example, the user does not have permission to read the authentication\_string column of any user, nor can they modify the user table directly.

- Apply table-level and column-level permissions in the same statement:

```
GRANT UPDATE(Name), DELETE ON world.country TO r_updater;
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Privileges available on the **column** level: SELECT, INSERT, UPDATE, and REFERENCES.

## Granting Roles to Users

The GRANT statement without an ON clause grants roles to MySQL user accounts and other roles.

- Example of granting roles

```
GRANT r_viewer, r_updater TO r_world;
GRANT r_viewer, r_updater TO kari@localhost WITH ADMIN OPTION;
```

- Statement clauses:

- List of roles to be granted
  - Example: r\_viewer, r\_updater
- Account or role to which you are granting the roles
- WITH ADMIN OPTION allows the user to grant the roles to other user accounts or revoke the roles from other user accounts.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Displaying GRANT Privileges

- Show your own account privileges with SHOW GRANTS:

```
mysql> SHOW GRANTS\G
***** 1. row *****
Grants for root@localhost: GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD,
SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE TEMPORARY
TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW,
CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE TABLESPACE, CREATE ROLE,
DROP ROLE ON *.* TO `root`@`localhost` WITH GRANT OPTION
***** 2. row *****
Grants for root@localhost: GRANT BACKUP_ADMIN,RESOURCE_GROUP_ADMIN,XA_RECOVER_ADMIN ON *.*
TO `root`@`localhost` WITH GRANT OPTION
***** 3. row *****
Grants for root@localhost: GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION
3 rows in set (0.00 sec)
```

- SHOW GRANTS displays the statements that re-create the privileges for the current user.
  - Synonym: SHOW GRANTS FOR CURRENT\_USER()



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

If the account can grant some or all of its privileges to other accounts, the output displays WITH GRANT OPTION at the end of each GRANT statement to which it applies.

SHOW GRANTS displays global privileges using:

- One line listing all granted static privileges, if there are any, including WITH GRANT OPTION if appropriate.
- One line listing all granted dynamic privileges for which GRANT OPTION is granted, if there are any, including WITH GRANT OPTION.
- One line listing all granted dynamic privileges for which GRANT OPTION is not granted, if there are any, without WITH GRANT OPTION.

## Displaying Privileges for Another User

Specify an account name to see privileges for that account:

```
mysql> SHOW GRANTS FOR kari@localhost;
+-----+
| Grants for kari@localhost |
+-----+
| GRANT USAGE ON *.* TO `kari`@`localhost` |
| GRANT SELECT (`host`, `user`) ON `mysql`.`user` TO `kari`@`localhost` |
| GRANT UPDATE, DELETE ON `world`.`city` TO `kari`@`localhost` |
| GRANT `r_updater`@`%`, `r_viewer`@`%` TO `kari`@`localhost` WITH ADMIN OPTION |
+-----+
4 rows in set (0.00 sec)
```

- The **ON** clauses in the preceding output display privileges at the global, database, and table levels.
- Column privileges are indicated by the parenthesized column list after a column-level privilege on a table-level grant.
- Lines without **ON** clause display roles granted to the user account.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

SHOW GRANTS statement allows you to view the privileges associated with roles for the user with the optional USING clause.

```
mysql> SHOW GRANTS FOR kari@localhost USING r_viewer, r_updater;
+-----+
| Grants for kari@localhost |
+-----+
| GRANT USAGE ON *.* TO `kari`@`localhost` |
| GRANT SELECT ON `world`.* TO `kari`@`localhost` |
| GRANT SELECT (`host`, `user`) ON `mysql`.`user` TO `kari`@`localhost` |
| GRANT UPDATE, DELETE ON `world`.`city` TO `kari`@`localhost` |
| GRANT UPDATE (`Name`), DELETE ON `world`.`country` TO `kari`@`localhost` |
| GRANT `r_updater`@`%`, `r_viewer`@`%` TO `kari`@`localhost` WITH ADMIN OPTION |
+-----+
6 rows in set (0.00 sec)
```

## Displaying Privileges for a Role

Specify a role name to see privileges for that role:

```
mysql> SHOW GRANTS FOR r_updater;
+-----+
| Grants for r_updater@% |
+-----+
| GRANT USAGE ON *.* TO `r_updater`@`%` |
| GRANT UPDATE (`Name`), DELETE ON `world`.`country` TO `r_updater`@`%` |
+-----+
2 rows in set (0.00 sec)
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Revoking Account Privileges

- Use the `REVOKE` statement to revoke privileges and roles from user accounts and roles.
- The `REVOKE` statement's syntax has the following clauses:
  - `REVOKE` keyword: Specifies the list of privileges or roles to be revoked
  - `ON` clause: Indicates the level at which privileges are to be revoked
    - Not required when revoking roles
  - `FROM` clause: Specifies the account name or role name
- Use the `SHOW GRANTS` statement before issuing `REVOKE` to determine which privileges and roles to revoke and then again afterward to confirm the result.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## REVOKE: Examples

- Assume that Amon has SELECT, DELETE, INSERT, and UPDATE privileges on the world database.

- You want to change the account so that he has SELECT access only.

```
REVOKE DELETE, INSERT, UPDATE ON world.* FROM 'Amon'@'localhost';
```

- Revoke Jan's ability to grant to other users any privileges that he holds for the world database, by revoking the GRANT OPTION privilege from his account.

```
REVOKE GRANT OPTION ON world.* FROM 'Jan'@'localhost';
```

- Assume that the r\_dev role has the CREATE, DROP, SELECT, DELETE, INSERT, and UPDATE privileges on the world database.

- You want to remove all DDL privileges from the role.

```
REVOKE CREATE, DROP ON world.* FROM r_dev;
```

- Revoke r\_updater role from Kari user account.

```
REVOKE r_updater FROM kari@localhost;
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## REVOKE: Examples

Revoke all privileges, including granting privileges to others:

- Remove all privileges held by Sasha's account (at any level), by revoking ALL PRIVILEGES and GRANT OPTION from her account.

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'Sasha'@'localhost';
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Partial Revoke

- Is available as of MySQL 8.0.16
- Is enabled when the `partial_revokes` system variable is set to `ON` the default is `OFF`
- Allows privileges to be granted globally except for certain schemas

```
mysql> CREATE USER u1 IDENTIFIED BY 'P@ssW0rd';
mysql> GRANT SELECT ON *.* TO u1;
mysql> REVOKE SELECT ON mysql.* FROM u1;
```

- The `REVOKE` will fail if the `partial_revokes` variable is not enabled.
- Use `GRANT` to remove the schema level restriction.

```
mysql> GRANT SELECT ON mysql.* TO u1;
```

- Cannot be disabled (`partial_revokes=OFF`) if any account has privilege restrictions
- Has conditions:
  - No wildcard characters (`_` or `%`) in the schema name
  - Apply to the schema level only, not the table, column, or routine levels



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## User Privilege Restrictions

- You cannot explicitly deny access to a specific user on a specific object.
  - Except using the partial revoke feature to revoke privileges on the schema level, which have been granted on the global level
- You cannot associate a password with a specific object such as a database, table, or routine.
- You cannot grant row-level privileges with the MySQL privilege system.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

You can use stored routines and views to approximate row-level permissions. Use conditional expressions in your SQL code, which compare an access control list in a table with the output of the CURRENT\_USER() function.

## Quiz



Which of the following statements prevent the kari@localhost user from logging in?

- a. ALTER USER kari@localhost ACCOUNT LOCK
- b. ALTER USER kari@localhost PASSWORD EXPIRE
- c. DROP USER kari@localhost
- d. REVOKE USAGE FROM kari@localhost



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

**Answer: a, c**

## Topics

- MySQL Privilege System
- Creating and Modifying User Accounts and Roles
- Configuring Passwords and Account Expiration
- Authentication Plugins
- Granting Permissions
- **Activating Roles**
- Grant Tables



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Using Role Privileges

- Roles must be activated before users can use the privileges granted to the roles.
- Roles can be activated on:
  - Server level
  - User level
  - Session level
- Users can only activate the roles that have been granted to them.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Activating Roles at Server Level

- Set `activate_all_roles_on_login` system variable to:
  - ON: The server activates all roles granted to each account at login time.
  - OFF (default value): The server activates the default roles specified with `SET DEFAULT ROLE`, if any, at login time.
- Applies only at login time and at the beginning of execution for stored programs and views that execute in definer context.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Activating Roles at User Level

- The `SET DEFAULT ROLE` statement defines which roles become active when the user connects to the server.

```
SET DEFAULT ROLE r_viewer, r_updater TO kari@localhost;
SET DEFAULT ROLE ALL TO kari@localhost, Jan@localhost;
```

- Statement clauses:
  - Roles to be activated: a list of roles or a specifier `ALL` or `NONE`
  - One or more user accounts
- Alternatively, use the `DEFAULT ROLE` clause in `CREATE USER` or `ALTER USER` statements.

```
ALTER USER kari@localhost DEFAULT ROLE r_viewer, r_updater;
ALTER USER kari@localhost DEFAULT ROLE ALL;
```

- User-level default roles are stored in the `mysql.default_roles` grant table.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Activating Roles at Session Level

- Use `SET ROLE` statement to modify the list of active roles in the current session. It accepts a list of roles or one of the following role specifiers:
  - `DEFAULT`: Activate the account default roles.
  - `NONE`: Disable all roles.
  - `ALL`: Activate all roles granted to the account.
  - `ALL EXCEPT`: Activate all roles granted to the account except those named.

```
SET ROLE ALL;
SET ROLE r_viewer, r_updater;
```

- Use `CURRENT_ROLE()` function to determine which roles are active in the current session.

```
SELECT CURRENT_ROLE();
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Mandatory Roles

Mandatory roles:

- Are automatically granted to every user
- Are configured using the `mandatory-roles` system variable
- Must be activated before the privileges take effect
- Do not change the grant tables
- Cannot be revoked or dropped using `REVOKE`, `DROP ROLE`, or `DROP USER` statements

Set `mandatory-roles` variable to a comma-separated list of role names. Example:

```
SET PERSIST mandatory_roles = ``role1``@``%``,role2,role3@localhost';
```



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Topics

- MySQL Privilege System
- Creating and Modifying User Accounts and Roles
- Configuring Passwords and Account Expiration
- Authentication Plugins
- Granting Permissions
- Activating Roles
- **Grant Tables**



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Grant Tables

- MySQL reads the grant tables from the `mysql` database into memory at startup and bases all access control decisions on those tables.
- Tables correspond to privilege levels:

| Table name                 | Contents and Privileges                             |
|----------------------------|-----------------------------------------------------|
| <code>user</code>          | User accounts and roles and global-level privileges |
| <code>global_grants</code> | Dynamic global privileges                           |
| <code>db</code>            | Database-level privileges                           |
| <code>tables_priv</code>   | Table-level privileges                              |
| <code>columns_priv</code>  | Column-level privileges                             |
| <code>role_edges</code>    | Roles granted to users and other roles              |



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

### Other grant tables:

- `procs_priv`: Stored procedure and function privileges
- `proxies_priv`: Proxy-user privileges
- `default_roles`: Default user roles
- `password_history`: Password change history

Do not modify the grant tables directly using DML statements such as `INSERT`, `UPDATE`, and `DELETE`. Instead, use the privilege management statements such as `CREATE USER`, `ALTER USER`, `GRANT`, and `REVOKE` statements to change the user credentials and privileges.

## Grant Table Contents

- The `user` table contains a record for each account known to the server, as well as its global privileges.
  - It also indicates other information about the account, such as:
    - Any resource limits that it is subject to
    - Whether client connections that use the account must be made over a secure connection using SSL
  - Every account must have a `user` table record; the server determines whether to accept or reject each connection attempt by reading the contents of that table.
- Each account also has records in the other grant tables if it has privileges at a level other than the global level.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Use of Grant Tables

- Each grant table has `host` and `user` columns to identify the accounts to which its records apply.
  - During connection attempts, the server determines whether a client can connect.
    - Using the `plugin` and `authentication_string` columns
  - After connection, the server determines access privileges for each statement.
    - The `user.%_priv` fields and the `db` and `%_priv` tables contain grant information.
    - The `User` and `Host` fields in each grant table identify the account.
- The MySQL installation process creates the grant tables.
  - Grant tables use the InnoDB storage engine.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Before MySQL 8.0, grant tables used the MyISAM storage engine and were non-transactional.

## Effecting Privilege Changes

- MySQL maintains in-memory copies of grant tables to avoid the overhead of accessing on-disk tables.
  - Avoid modifying a user account directly in the grant tables using DML statements.
    - Mistakes can lock you out of the system.
    - If you modify the grant tables directly, reload the tables explicitly by issuing a `FLUSH PRIVILEGES` statement.
- Account modification statements such as `GRANT`, `REVOKE`, `SET PASSWORD`, and `RENAME USER` apply changes to both the grant tables and the in-memory table copies.
- Changes to global privileges and passwords apply only to subsequent connections of that account.
- Changes to database-level privileges apply after the client's next `USE db_name` statement.
- Changes to role, table, column and routine privileges apply immediately.



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Summary



In this lesson, you should have learned how to:

- Create user accounts and roles
- Design a permissions structure
- Control user and role permissions
- Grant access to system operations
- Use authentication plugins
- Expire accounts manually and automatically



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Practices

- 6-1: Quiz—User Management
- 6-2: Creating Users and Roles
- 6-3: Granting Permissions



ORACLE®

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.