



Integrated Cloud Applications & Platform Services

MySQL for Database Administrators

Student Guide - Volume I

D61762GC50

Edition 5.0 | June 2019 | D106724

Learn more from Oracle University at education.oracle.com



Authors

KimSeong Loh

Mark Lewin

Technical Contributors and Reviewers

Frederic Descamps

Hananto Wicaksono

Igor Ilyin

Mirko Ortensi

Editors

Adrita Biswas

Aju Kumar

Raj Kumar

Moushmi Mukherjee

Graphic Editors

Kavya Bellur

Pushparaj Kundar

Publishers

Pavithran Adka

Veena Narasimhan

Copyright © 2019, 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
- A Modern Database for the Digital Age 1-8
- High Scalability 1-9
- MySQL Enterprise Edition 1-10
- Oracle Premier Support for MySQL 1-11
- MySQL and Oracle Integration 1-12
- MySQL as a Service 1-13
- MySQL Websites 1-14
- Community Resources 1-15
- Oracle University: MySQL Training 1-16
- MySQL Certification 1-17
- Summary 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
Installing MySQL in Oracle Public Cloud	2-24
Oracle Public Cloud Concepts	2-25
MySQL Cloud Service	2-26
MySQL Cloud Service Management	2-27
Creating a New MySQL Cloud Service Instance	2-28
Administering MySQL Cloud Service Instances	2-29
MySQL Applications in Oracle Public Cloud	2-30
Topics	2-31
Linux MySQL Server Installation Directories	2-32
Windows MySQL Server Installation Directory	2-33
MySQL Programs	2-34
mysqld: MySQL Server Process	2-35
Server Helper Programs	2-36
How the Server Helper Programs Interact	2-37
Installation Programs	2-38
mysql_secure_installation	2-39
Utility Programs	2-40
mysql_config_editor	2-41
.mylogin.cnf Format	2-42
Login Paths	2-43
Quiz	2-44
Command-Line Client Programs	2-45
Launching Command-Line Client Programs	2-46
Topics	2-47
Configuring Mandatory Access Control	2-48
SELinux Example	2-49
AppArmor: Example	2-50
Changing the root Password	2-51
Using mysqladmin to Change the root Password	2-52
Topics	2-53
Starting and Stopping MySQL	2-54
Stopping MySQL with mysqladmin	2-55
MySQL Service Files	2-56
Starting and Stopping MySQL on Windows	2-57
Starting and Stopping MySQL on Windows: MySQL Notifier	2-58
Topics	2-59

Upgrading MySQL 2-60
Reading Release Notes 2-61
MySQL Shell Upgrade Checker Utility 2-62
Selecting an Upgrade Method 2-63
mysql_upgrade 2-64
Quiz 2-65
Summary 2-66
Practices 2-67

3 Understanding MySQL Architecture

Objectives 3-2
Topics 3-3
Architecture 3-4
Client/Server Connectivity 3-5
MySQL Server 3-6
Terminology: Server and Host 3-7
Server Process 3-8
Topics 3-9
Connection Layer 3-10
Communication Protocols 3-11
Local and Remote Communications Protocol: TCP/IP 3-12
Local Communication Protocol in Linux: Socket 3-13
MySQL and localhost 3-14
Local Communications Protocols in Windows: Shared Memory and Named Pipes 3-15
SSL by Default 3-16
Connection Threads 3-17
Topics 3-18
SQL Layer 3-19
SQL Layer Components 3-20
SQL Statement Processing 3-21
Quiz 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 Features 3-30
InnoDB Storage Engine: Highlights 3-31
MyISAM Storage Engine 3-32

MEMORY Storage Engine 3-33
ARCHIVE Storage Engine 3-34
BLACKHOLE Storage Engine 3-35
How MySQL Uses Disk Space 3-36
Data Directory Files 3-37
Topics 3-38
What Is a Data Dictionary? 3-39
Role of the Data Dictionary 3-40
Types of Metadata 3-41
Data Dictionary in Earlier Versions of MySQL 3-42
Transactional Data Dictionary in MySQL 8 3-43
Transactional Data Dictionary: Features 3-44
Serialization of the Data Dictionary 3-45
Topics 3-46
InnoDB Tablespaces 3-47
InnoDB System Tablespace 3-48
Options That Control Tablespaces 3-49
File-per-Table Tablespaces 3-50
General Tablespaces 3-51
Choosing Between File-per-Table and General Tablespaces 3-52
Locating Tablespaces Outside the Data Directory 3-53
Temporary Tablespaces 3-54
Topics 3-55
Redo Logs 3-56
Undo Logs 3-58
Undo Tablespaces 3-60
Temporary Table Undo Log 3-61
Quiz 3-62
Topics 3-63
How MySQL Uses Memory 3-64
Global Memory 3-66
Session Memory 3-67
Log Files and Buffers 3-68
InnoDB Buffer Pool 3-69
Configuring the Buffer Pool 3-70
Topics 3-71
MySQL Plugin Interface 3-72
Summary 3-73
Practices 3-74

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
 - Changing Variable Values 4-28
 - Dynamic System Variables 4-29
 - System Variable Types 4-30
 - Displaying System Variables 4-31
 - Viewing Variables with Performance Schema 4-32
 - Persisting Global Variables 4-33
 - 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
Log File Usage Matrix 5-6
Enabling Logs 5-7
General Query Log 5-9
General Query Log: Example 5-10
Slow Query Log 5-11
Slow Query Log: Logging Administrative and Replicated Statements 5-12
Filtering Slow Query Log Events 5-13
Slow Query Log: Example 5-14
Viewing the Slow Query Log with mysqldumpslow 5-15
mysqldumpslow: Example 5-16
Specifying TABLE or FILE Log Output 5-17
Log File Rotation 5-18
Flushing Logs 5-19
Topics 5-20
Status Variables 5-21
Displaying Status Information 5-22
Monitoring Status with mysqladmin 5-23
Quiz 5-24
Topics 5-25
Performance Schema 5-26
Performance Schema Table Groups 5-27
Configuring Performance Schema 5-28
Performance Schema Setup Tables 5-29
Performance Schema Instruments 5-30
Top-Level Instrument Components 5-31
Accessing Performance Schema Metrics 5-32
The sys Schema 5-33
Using the sys Schema Example 1 5-34
Using the sys Schema: Example 2 5-37
Topics 5-38
Configuring MySQL Enterprise Audit 5-39

Installing MySQL Enterprise Audit 5-40
Audit Log File Configuration 5-41
Audit Log File Contents 5-42
Audit Records 5-43
Audit Log Attributes 5-44
Topics 5-45
MySQL Enterprise Monitor 5-46
Installing MySQL Enterprise Monitor 5-47
Installing the Service Manager 5-48
Post-Installation Configuration 5-50
Installing Agents 5-51
MySQL Enterprise Monitor: Managing Multiple Servers 5-52
MySQL Enterprise Monitor: Timeseries Graphs 5-53
MySQL Enterprise Monitor: Advisors 5-54
MySQL Enterprise Monitor: Events 5-55
Topics 5-56
SHOW PROCESSLIST 5-57
Killing Processes 5-58
Limiting User Activity 5-59
Setting Resource Limits 5-60
Resetting Limits to Default Values 5-61
Quiz 5-62
Summary 5-63
Practices 5-64

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
Expiring Passwords Manually	6-20
Configuring Password Expiration	6-21
Changing Expired Passwords	6-22
Quiz	6-23
Topics	6-24
Pluggable Authentication	6-25
Cleartext Client-Side Authentication Plugin	6-26
Loadable Authentication Plugins	6-27
PAM Authentication Plugin	6-28
Configuring the PAM Authentication Plugin	6-29
Creating Users that Authenticate with PAM	6-30
Creating PAM Proxied Users	6-31
Logging In with PAM Accounts	6-32
Topics	6-33
Authorization	6-34
Determining Appropriate User Privileges	6-35
Privilege Scope	6-36
Granting Administrative Privileges	6-37
Dynamic Privileges	6-38
Special Privileges	6-39
GRANT Statement	6-40
Granting Permissions on Columns	6-41
Granting Roles to Users	6-42
Displaying GRANT Privileges	6-43
Displaying Privileges for Another User	6-44
Displaying Privileges for a Role	6-45
User Privilege Restrictions	6-46
Revoking Account Privileges	6-47
REVOKE: Examples	6-48
Quiz	6-50
Topics	6-51
Using Role Privileges	6-52
Activating Roles at Server-level	6-53
Activating Roles at User-level	6-54
Activating Roles at Session-level	6-55
Mandatory Roles	6-56
Topics	6-57
Grant Tables	6-58

Grant Table Contents	6-59
Use of Grant Tables	6-60
Effecting Privilege Changes	6-61
Summary	6-62
Practices	6-63

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
Using One or More Firewalls	7-8
Preventing Network Security Risks	7-10
Securing MySQL in Private Networks	7-11
Topics	7-12
Secure Connections	7-13
Secure Connection: Overview	7-14
Generating a Digital Certificate	7-15
Server Security Defaults	7-16
SSL Is Enabled by Default with MySQL Clients	7-17
Disabling SSL on MySQL Server	7-18
Setting Client Options for Secure Connections	7-19
Client --ssl-mode Option: Example	7-20
Setting the Permitted Versions for SSL/TLS for the Server	7-21
Setting the Permitted Versions for SSL/TLS for the Client	7-22
Setting the Cipher to Use for Secure Connections	7-23
Global System Variable and Session Status Variables for Ciphers	7-24
Cipher System and Status Variables: Example 1	7-25
Cipher System and Status Variables: Example 2	7-26
Setting Client SSL/TLS Options by User Account	7-27
Generating a Digital Certificate	7-28
SSL Server Variables for Digital Certificates	7-29
SSL Client Options for Digital Certificates	7-30
Securing a Remote Connection to MySQL	7-31
Quiz	7-32
Topics	7-33
Preventing MySQL Password Security Risks	7-34
How Attackers Derive Passwords	7-35
Password Validation Plugins and Components	7-36

Other Variables for the Password Validation Component	7-37
Installing the Password Validation Component	7-38
Validate Password Component Variables	7-39
Changing the Default Password Validation Variables	7-40
Other Password Considerations	7-41
Locking an Account	7-42
Pluggable Authentication	7-43
Preventing Application Password Security Risks	7-44
Connection-Control Plugin	7-45
Installing the Connection-Control Plugin	7-46
Monitoring Connection Failures	7-47
Using the CONNECTION_CONTROL_FAILED_LOGIN_ATTEMPTS Plugin	7-48
Quiz	7-49
Topics	7-50
Limiting Operating System Usage	7-51
Limiting Operating System Accounts	7-52
Operating System Security	7-53
MySQL Service Security	7-54
File System Security	7-55
Preventing File System Security Risks	7-56
Topics	7-57
Protecting Your Data from SQL Injection Attacks	7-58
SQL Injection: Example	7-59
Detecting Potential SQL Injection Attack Vectors	7-60
Preventing SQL Injection Attacks	7-61
Topics	7-62
MySQL Enterprise Firewall	7-63
Enterprise Firewall Plugins	7-64
Enterprise Firewall Database Components	7-65
Installing MySQL Enterprise Firewall	7-66
Registering Accounts with the Firewall	7-67
Training the Firewall	7-68
Statement Digests	7-69
Enabling Firewall Protection	7-70
Disabling the Firewall	7-71
Monitoring the Firewall	7-72
Quiz	7-73
Summary	7-74
Practices	7-75

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
How MySQL Locks Rows	8-37
Identifying Lock Contention	8-38
InnoDB Table Locks	8-39
InnoDB Row Locks	8-40
Troubleshooting Locks with SHOW PROCESSLIST	8-41

SHOW PROCESSLIST: Example 8-42
Monitoring Data Locks with Information Schema and Performance Schema 8-43
The Information Schema INNODB_TRX View 8-44
The Performance Schema data_locks Table 8-45
The Performance Schema data_lock_waits Table 8-47
sys.innodb_lock_waits View 8-48
sys.innodb_lock_waits: Example Query 8-49
Metadata Locks 8-50
Reading the metadata_locks Table 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
Index Types 9-19
MySQL Enterprise Monitor Query Analyzer 9-20
Query Response Time Index 9-21
Query Analyzer User Interface 9-22
Topics 9-23
Creating Indexes to Improve Query Performance 9-24
Creating and Dropping Indexes on Existing Tables 9-25
Displaying Indexes with SHOW INDEXES FROM 9-26
Topics 9-27

Maintaining InnoDB Index Statistics	9-28
Automatically Updating Index Statistics	9-29
Using ANALYZE TABLE	9-30
Rebuilding Indexes	9-31
mysqlcheck Client Program	9-32
Invisible Indexes	9-33
Histograms	9-34
Example: The Query	9-35
Example: Creating a Histogram	9-36
Example: The Query with Histogram Data	9-37
Quiz	9-38
Summary	9-39
Practices	9-40

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
MySQL Enterprise Backup 11-5
MySQL Enterprise Backup: Storage Engines 11-6
MySQL Enterprise Backup: InnoDB Files 11-7
MySQL Enterprise Backup: Non-InnoDB Files 11-8
MySQL Enterprise Backup: Use Cases 11-9
MySQL Enterprise Backup: Basic Usage 11-10
Backup Process 11-11
Restoring a Backup with MySQL Enterprise Backup 11-12
Using the copy-back Command 11-13
MySQL Enterprise Backup: Single-File Backups 11-14
MySQL Enterprise Backup: Restoring Single-File Backups 11-15
Basic Privileges Required for MySQL Enterprise Backup 11-16
Granting Required Privileges 11-17
Quiz 11-18
mysqldump and mysqlpump 11-19
mysqldump 11-20
Ensuring Data Consistency with mysqldump 11-21
mysqldump Options for Creating Objects 11-22
mysqldump Options for Dropping Objects 11-23
mysqldump General Options 11-24
Restoring mysqldump Backups 11-25
Using mysqlimport 11-26
Privileges Required for mysqldump 11-27
Privileges Required for Reloading Dump Files 11-28
mysqlpump 11-29
Specifying Objects to Back Up with mysqlpump 11-30
Parallel Processing with mysqlpump 11-31
Quiz 11-32
Topics 11-33
Raw InnoDB Backups: Overview 11-34
Portability of Raw Backups 11-35
Raw InnoDB Backup Procedure 11-36

Recovering from Raw InnoDB Backups	11-37
Using Transportable Tablespaces for Backup	11-38
Transportable Tablespaces: Copying a Table to Another Instance	11-39
Raw MyISAM and ARCHIVE Backups	11-40
Raw MyISAM and ARCHIVE Backup Procedure	11-41
Recovering from Raw MyISAM or Archive Backups	11-42
LVM Snapshots	11-43
Creating LVM Snapshots	11-44
LVM Backup Procedure	11-45
LVM Backup: Example	11-46
Backing Up Log and Status Files	11-47
Topics	11-48
Replication as an Aid to Backup	11-49
Backing Up from a Replication Slave	11-50
Backing Up from Multiple Sources to a Single Server	11-51
Processing Binary Log Contents	11-52
Selective Binary Log Processing	11-53
Point-in-Time Recovery	11-54
Using mysqlbinlog for Point-in-Time Recovery	11-55
Configuring MySQL for Restore Operations	11-56
Quiz	11-57
Summary	11-58
Practices	11-59

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
Asynchronous Replication	12-32
Semisynchronous Replication	12-33
Advantages and Disadvantages of Semisynchronous Replication	12-34
Enabling Semisynchronous Replication	12-35
Quiz	12-36
Binary Logging	12-37
Binary Log Formats	12-38
Row-Based Binary Logging	12-39
Statement-Based Binary Logging	12-40
Mixed Format Binary Logging	12-41
Replication Logs	12-42
Crash-Safe Replication	12-43
Multisource Replication	12-44
Configuring Multisource Replication for a GTID-Based Master	12-45
Configuring Multisource Replication for a Binary Log-Based Master	12-46
Controlling Slaves in a Multisource Replication Topology	12-47
Summary	12-48
Practices	12-49

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
Multi-Threaded 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 Multisource 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
Use Cases	14-11
Group Replication: Requirements and Limitations	14-12
Quiz	14-13
Topics	14-14
MySQL Shell (mysqlsh)	14-15

Using MySQL Shell to Execute a Script	14-16
MySQL Router (mysqlrouter)	14-17
Topics	14-18
Deployment Scenarios	14-19
Deploying Sandbox Instances and Creating the Cluster	14-20
Production Deployment	14-21
Connecting Clients to the Cluster	14-22
Topics	14-23
Managing Sandbox Instances	14-24
Checking the Status of a Cluster	14-25
Viewing the Structure of a Cluster	14-26
Rescanning a Cluster	14-27
Removing Instances from the Cluster	14-28
Customizing a MySQL InnoDB Cluster	14-29
Checking the State of an Instance	14-30
Rejoining a Cluster	14-31
Restoring Quorum Loss	14-32
Recovering the Cluster from a Major Outage	14-33
Enabling and Disabling Writes with super_read_only	14-34
Quiz	14-35
Securing a Cluster	14-36
Securing Instances	14-37
Creating a Server Whitelist	14-38
Dissolving a Cluster	14-39
Summary	14-40
Practices	14-41

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®



MySQL™

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide

Objectives



After completing this lesson, you should be able to:

- Describe the course goals
- List MySQL products and professional services
- State the benefits of using MySQL in the cloud
- Access MySQL information and services from Oracle websites and community resources
- Find information about MySQL courses and certification options

ORACLE®

Copyright © 2019, 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



Copyright © 2019, 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



Copyright © 2019, 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 © 2019, 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



Copyright © 2019, 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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

A Modern Database for the Digital Age

Digital Disruptors and Large Enterprises Rely on MySQL to Innovate



Square

TESLA

NETFLIX



Dropbox



LinkedIn

Pinterest

YouTube

Booking.com

zendesk

ORACLE

Copyright © 2019, 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, and so do large enterprises driving 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.

High Scalability



Mobile Network Supporting Over 800 Million Subscribers



2 Billion Events/Day for Booking.com



IDs Processed for 1 Billion Citizens



2.3 Billion Active Users



100 TB of User Data for PayPal



850 Million Candy Crush Game Plays/Day

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

- MySQL Cluster powers a mobile network in Asia counting over 800 million subscribers.
- A government, also in Asia, has processed over one billion IDs for its citizens.

MySQL Enterprise Edition

Advanced Features	Management Tools	Support
Security Network Access Control MySQL Enterprise Authentication MySQL Enterprise Audit MySQL Enterprise Encryption/Transparent Data Encryption (TDE) MySQL Enterprise Firewall MySQL Enterprise Masking and De-identification High Availability MySQL Enterprise HA Scalability MySQL Enterprise Thread Pool	Monitoring MySQL Enterprise Monitor Backup MySQL Enterprise Backup Development MySQL Connectors Administration and Migration MySQL Workbench	Technical and Consultative Support – Oracle Premier Support for MySQL Oracle Certifications – Oracle Certified MySQL Database Administrator – Oracle Certified MySQL Developer



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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 High Availability

MySQL Enterprise High Availability enables you to meet the availability requirements of even the most demanding, mission-critical applications. MySQL InnoDB Cluster provides native high availability with built-in group membership management, data consistency guarantees, conflict detection and handling, node failure detection, and database failover-related operations, all without the need for manual intervention or custom tooling.

MySQL Security

- **Network Access Control:** Restrict connections to your MySQL instances.
- **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 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.

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 © 2019, Oracle and/or its affiliates. All rights reserved.

MySQL and Oracle Integration

MySQL integrates into the Oracle Environment



ORACLE

Copyright © 2019, 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 as a Service

- Powerful union of MySQL Enterprise Edition and Oracle Cloud Infrastructure
 - Amazing underlying software and high-performance hardware
- Self-managed and easy to use
 - Automates most common DBA tasks: backups, patches, updates, replication configuration, etc.
 - Web console, REST API, CLI, SDKs, and seamless integration
- Elasticity and high availability
 - Replication across different physical locations
- Security and compliance
 - Ready-to-use advanced security options and compliant with the most demanding enterprise regulations



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle MySQL Cloud Service enables you to rapidly, securely and cost-effectively deploy modern applications powered by MySQL.

MySQL Cloud Service is built on Oracle Cloud Infrastructure, a second generation cloud infrastructure platform capable of running traditional multi-tiered enterprise applications, high-performance workloads, and modern serverless and container-based architectures.

MySQL as a Service (MySQLaaS) provides a service that:

- is simple to quickly provision a MySQL database with just few clicks
- provides tools to automate administration tasks
- is integrated with other Oracle Cloud Services for quick development/deployment
- is enterprise ready with Enterprise Edition features, management tools and 24x7 support

Because MySQL is developed at Oracle, it is the only public cloud to offer MySQL Enterprise Edition as a Database Service

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



Copyright © 2019, 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 Forums
- MySQL Community Slack
- 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

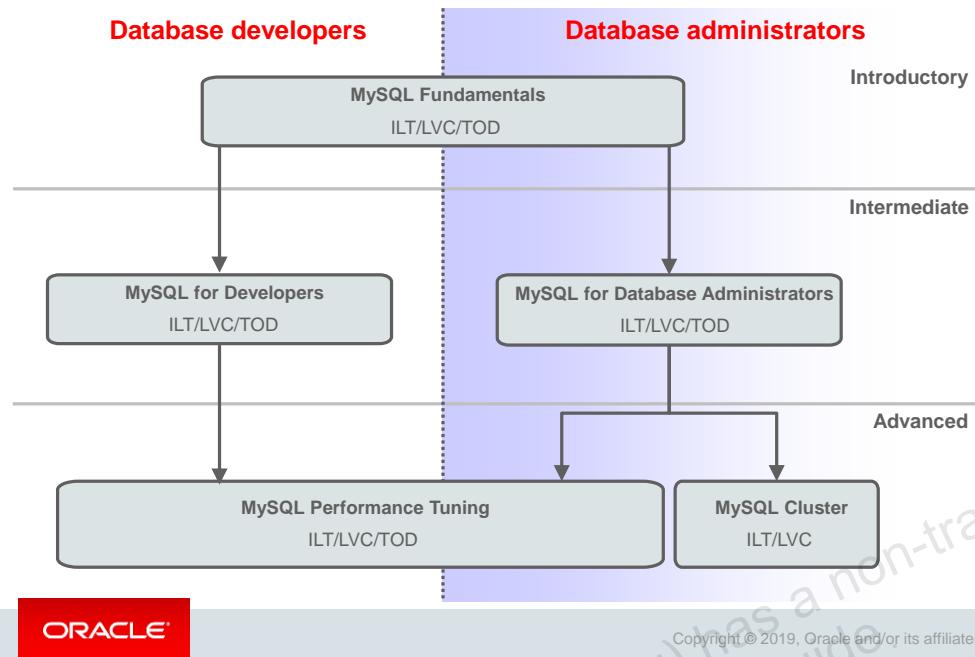


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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 Forums (<http://forums.mysql.com>)
- MySQL Community Slack (<https://mysqlcommunity.slack.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 © 2019, 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 instructor 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



Copyright © 2019, 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
- State the benefits of using MySQL in the cloud
- Access MySQL information and services from Oracle websites and community resources
- Find information about MySQL courses and certification options

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

2

Installing and Upgrading MySQL

ORACLE®



MySQL™

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide

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

ORACLE®

Copyright © 2019, 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 © 2019, 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.



Copyright © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

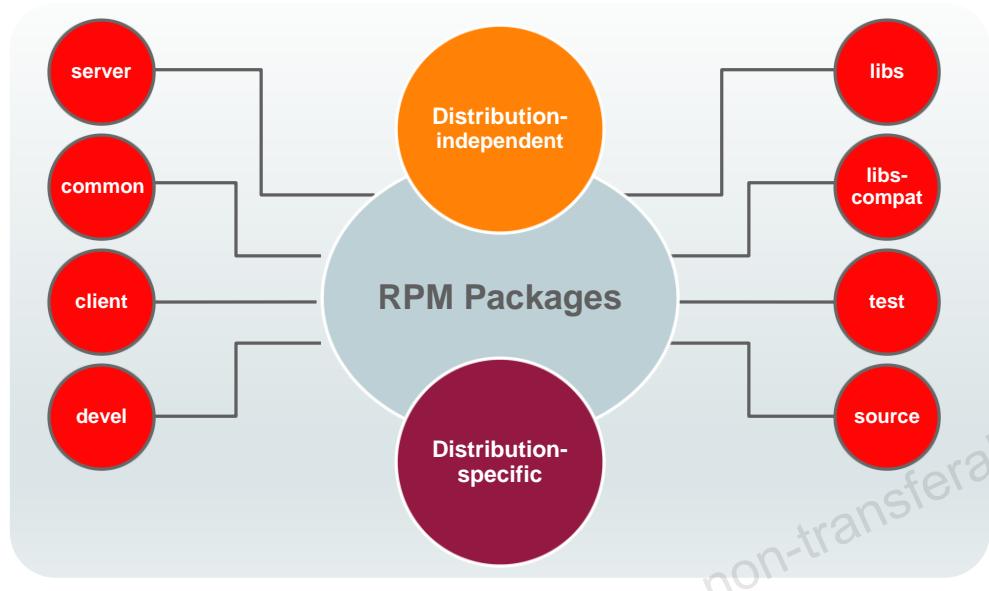
Installing Dependencies

On RPM machines that can reach repositories over the network or on a connected file system, 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 © 2019, 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:** Binaries, configuration, and databases for the MySQL server
- **Common:** Common files for server and client libraries
- **Client:** Client programs that connect to the database server
- **Devel:** Libraries needed when you compile MySQL programs
- **Libs:** Libraries used by many applications that connect to MySQL
- **Libs-compat:** Shared compatibility libraries for previous MySQL installations
- **Test:** MySQL test suite
- **Source:** MySQL source code

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 mysql start`, MySQL:
 - Creates the data directory and the default `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 log file (`/var/log/mysql/mysqld.log`)
 - You must change the password before you can use MySQL.



Copyright © 2019, 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 file system. 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 mysql start` – For Red Hat Enterprise Linux, Oracle Linux, CentOS, and Fedora systems
- `service mysqld start` – For SUSE Enterprise Linux Server

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 © 2019, Oracle and/or its affiliates. All rights reserved.

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 file system
- 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 © 2019, 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` packages also installs the packages for the components the server requires.



Copyright © 2019, 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 file system. 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-el7-1.noarch.rpm`.
- Install the file by using the `yum localinstall` command, for example:

```
yum localinstall mysql80-community-release-el7-1.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 © 2019, 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 `/etc/yum.repos.d/mysql-community.repo`:

- The following extract shows the default disabled configuration for the MySQL 5.7 repository:

```
# 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.

ORACLE

Copyright © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

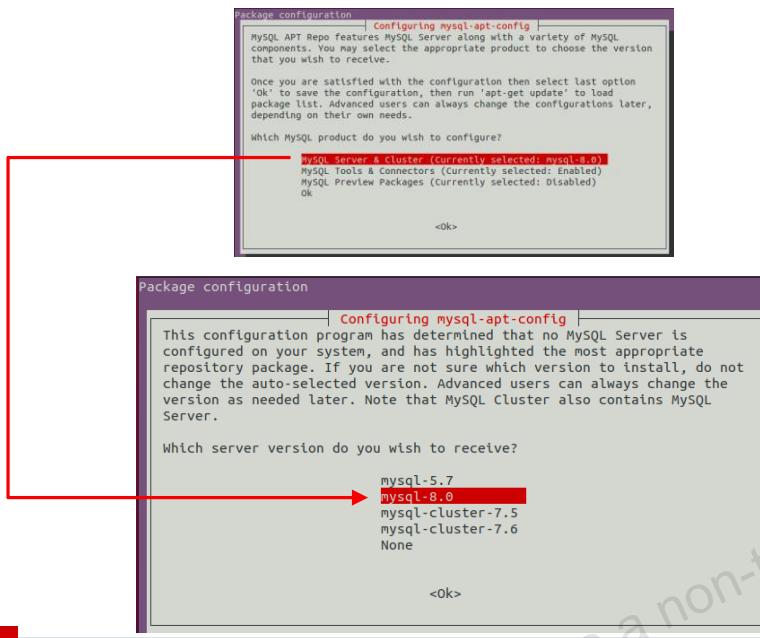
Repository Configuration

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 © 2019, 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 CD to update

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:

```
### 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
```

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

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

ORACLE

Copyright © 2019, 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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

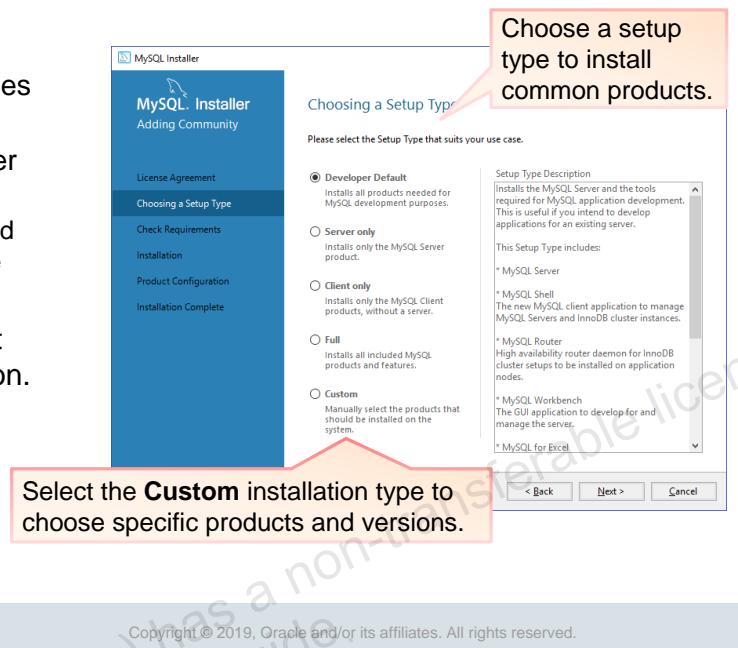
Included Products

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

- MySQL Workbench
- MySQL connectors (.Net / Python / ODBC / Java / C / C++)
- MySQL Notifier
- MySQL for Excel
- MySQL for Visual Studio
- MySQL Utilities
- MySQL Shell
- MySQL Router
- Samples, examples, and documentation

Installing on Windows: MySQL Installer

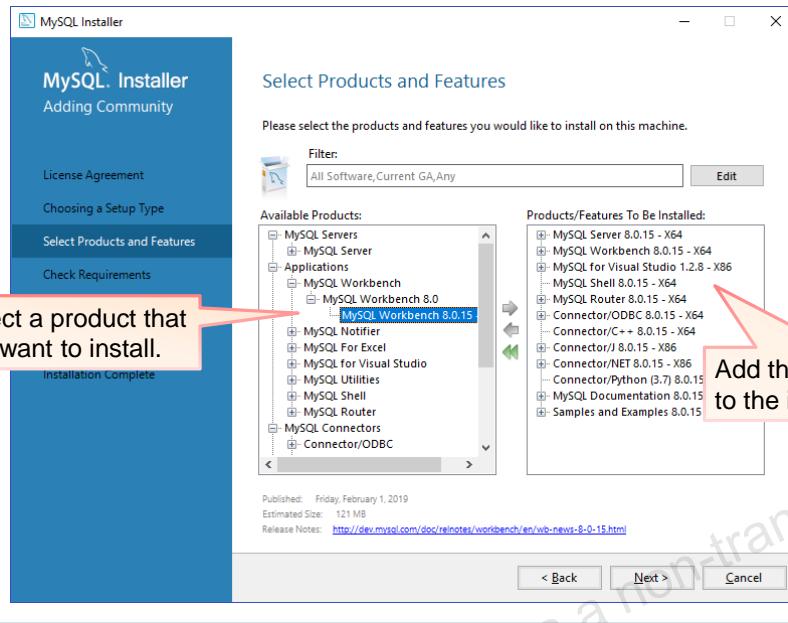
- Download the Full or Web installers:
 - **Full:** Contains all installation files in the downloaded file
 - **Web:** Contains only the installer
 - The installer downloads additional required files based on your selections during the installation process.
- Follow the Installer screens to select products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



ORACLE®

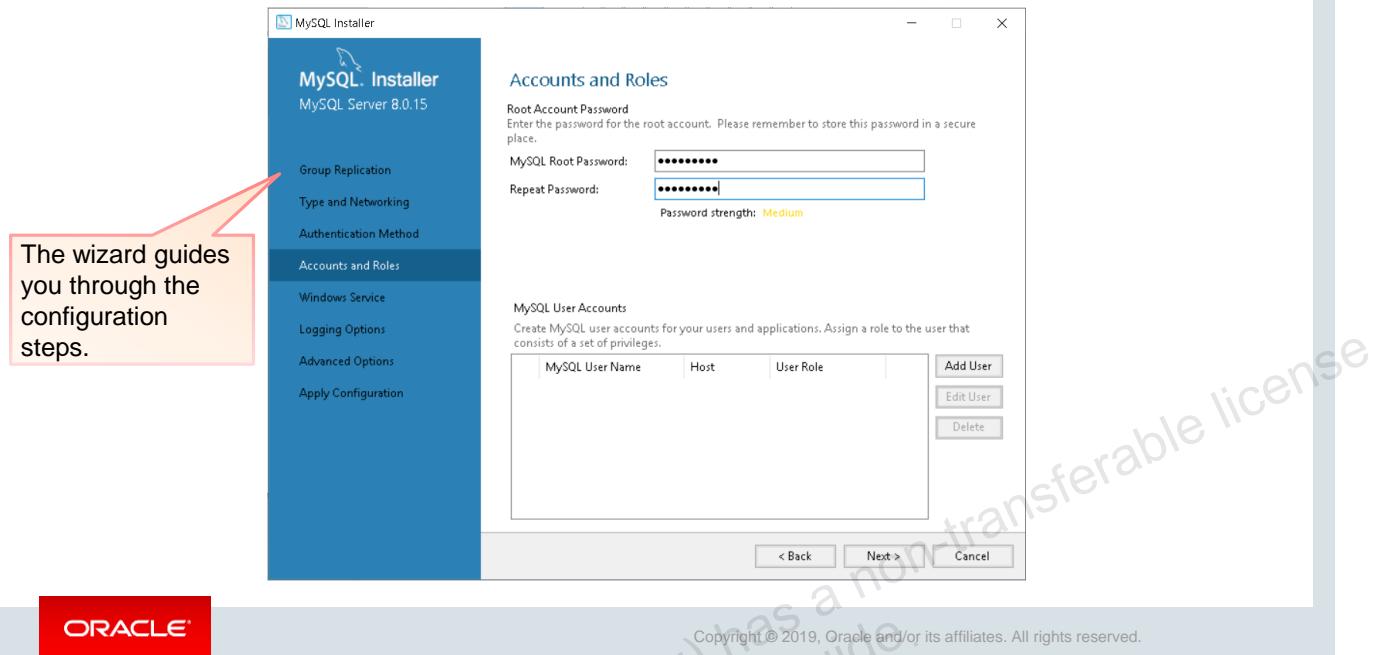
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Installing on Windows: Selecting Products and Features



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Installing on Windows: Product Configuration



The wizard guides you through the following configuration steps:

- Group Replication
- 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
```
 - 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 © 2019, 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 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



Copyright © 2019, 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 or another ANSI C++ compiler

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.
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/
# su - mysql
$ cd /usr/local/mysql-8.0.x/
$ tar xf ~/mysql-8.0.x-linux-glibc2.12-x86_64.tar.gz
```



Copyright © 2019, 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.

Installing MySQL from Binary Archive

3. Initialize the data directory:

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

4. Create the initial configuration file.

- Copy `my-default.cnf` to `/etc/my.cnf`.
- Edit the `datadir` setting to point to the data directory you initialized in step 3.
- Edit any other required settings.
 - Log file settings
 - TCP port

5. Optionally, populate time zone tables.

- Example:

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



Copyright © 2019, 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.

Installing MySQL in Oracle Public Cloud

Oracle Public Cloud offers Platform as a Service (PaaS).

- MySQL Cloud Service enables you to quickly provision, scale, back up, and patch MySQL instances on a fully managed scalable off-premises host.
- Host as many or as few instances as per your short- and long-term needs.
 - They are limited only by the amount of OCPUs and RAM in your account.
 - OCPUs can be distributed among your instances as your application demand changes.



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

An OCPU (Oracle CPU) is defined as the CPU capacity equivalent of one physical core of an Intel Xeon processor with Hyper-threading (two hardware execution threads) enabled.

Oracle Public Cloud Concepts

- Oracle Public Cloud account:
 - Enables specific cloud services and storage
- Identity domain:
 - Contains users within a cloud account and maps them to roles, with privileges over cloud services
- Cloud Services:
 - Enable provisioning, hosting, and feature support for IaaS, PaaS, SaaS
 - Examples: MySQL Cloud Service, Java Cloud Services, Application Container Cloud
- Compute shapes:
 - Quantity of OCPUs and RAM dedicated to each instance



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Compute shapes are named configurations of OCPUs and RAM. For example, the OC3 compute shape provides one OCPU and 7.5 GB RAM, and OC5m provides 16 OCPUs and 240 GB RAM.

Acronyms

- **IaaS:** Infrastructure as a service
- **PaaS:** Platform as a service
- **SaaS:** Software as a service

MySQL Cloud Service

MySQL Enterprise Edition is hosted on Oracle Infrastructure Cloud Services.

- Oracle Cloud provides the underlying robust and elastic infrastructure, networking, and storage.
- MySQL Cloud Service tooling automates and simplifies provisioning and administrative activities.
 - Provisioning, patching, backup, restore, and scaling
- Instances include MySQL Enterprise Edition features such as:
 - MySQL Enterprise Monitor
 - MySQL Enterprise Backup
 - Enterprise Thread Pool, Enterprise Audit, Enterprise Firewall
- The stack enables single-vendor support through the whole application.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

MySQL Cloud Service Management

Create MySQL Cloud Service instances easily through the web-based PaaS Service Manager (PSM) user interface.

- View any existing instances.
- Provision new instances by using the “Create Service” Wizard.
- Administer existing instances.
- Prerequisites:
 - Supported Browser
 - Any popular browser that has been updated recently
 - Account on Oracle Public Cloud, enabling MySQL Cloud Service and (optionally) Storage Cloud
 - SSH client for administration
 - Register your public key with the cloud instance.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can find the complete list of supported browsers for Oracle Cloud and related services at the following location: <https://docs.oracle.com/en/cloud/get-started/subscriptions-cloud/csgsg/web-browser-requirements.html>

Creating a New MySQL Cloud Service Instance

- Log in to your data center on Oracle Public Cloud.
- Connect to MySQL Cloud Service PSM UI using your identity domain credentials.
- Launch the Create Service Wizard, specifying:
 - A name and description for the new instance
 - An SSH public key for administration
 - A compute shape for the new instance
 - The initial database name and storage
 - The initial administration user credentials
 - Whether to configure MySQL Enterprise Monitor and its initial credentials
 - Whether to configure backups and their initial configuration and Storage Cloud credentials



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can upload your own SSH public key when you run the wizard. Alternatively, the wizard enables you to create a new public/private SSH key pair that you can download to your computer.

Administering MySQL Cloud Service Instances

- Easily create and delete throwaway or single-purpose instances.
- Create multiple instances up to your resource limit.
- Scale up by changing your instance's compute shape as your business and application demands change.
- Monitor your instances:
 - Perform on-demand healthchecks in the PSM UI.
 - Monitor your enterprise (on-premises and in the cloud) with MySQL Enterprise Monitor graphs, advisors, and alerts.
- Perform on-demand or automated backups.
 - MySQL Enterprise Backup is automatically installed.
- Patch MySQL with precheck and automated patching.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

- MySQL Enterprise Monitor is covered in the lesson titled “Monitoring MySQL.”
- MySQL Enterprise Backup is covered in the lesson titled “Performing Backups.”

MySQL Applications in Oracle Public Cloud

Oracle Public Cloud also offers Infrastructure as a Service (IaaS) and Software as a Service (SaaS) so that you can run server applications, such as MySQL client applications, in the cloud.

- Compute Cloud instances are managed IaaS virtual hosts.
 - Dedicated or multitenant infrastructure
 - Use host templates or upload your own virtual machine image.
- Application Containers make it easy to develop and deploy SaaS applications.
 - Cloud-native Java, PHP, Node.js development
 - Integrated life-cycle tools for continuous integration and deployment



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Cloud also offers DaaS (data as a service) and its own SaaS applications when you want to run Oracle applications, middleware, or analytics.

Compute instances have elastic compute capacity measured in OCPUs, where each OCPU is equivalent to a physical CPU core with two hardware execution threads (vCPUs). You can easily scale the number of OCPUs assigned to a Compute instance without rebuilding the image.

For more information about Oracle Compute Cloud, see <https://cloud.oracle.com/compute>.

Topics

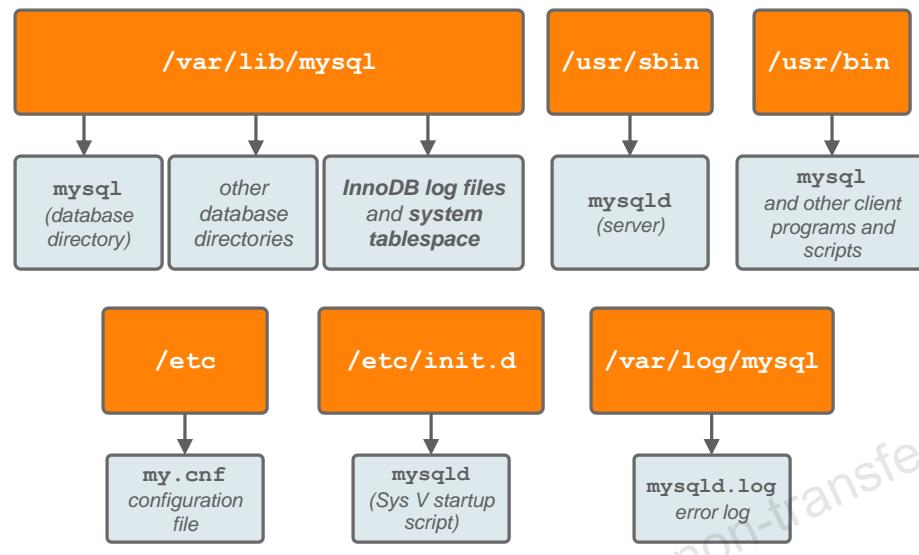
- Installing MySQL
- **Installed Files and Directories**
- Initial Configuration
- Starting and Stopping MySQL
- Upgrading MySQL



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Linux MySQL Server Installation Directories



ORACLE

Copyright © 2019, 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 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.

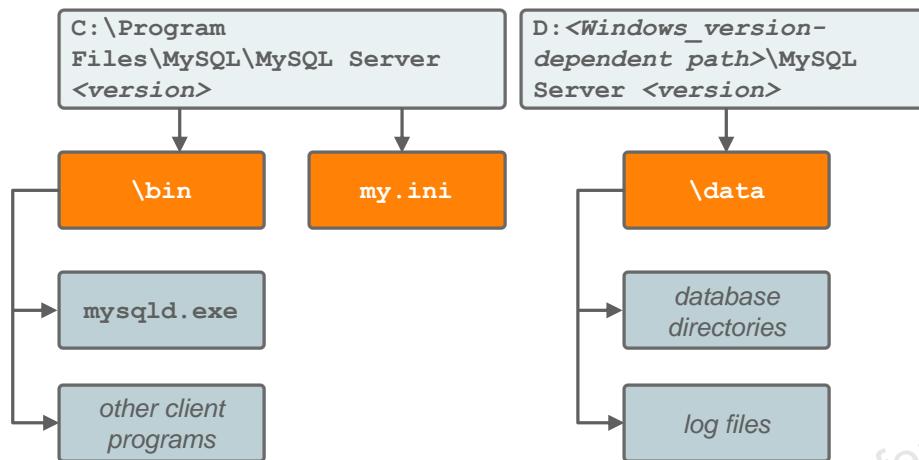
Base Directory

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

Other Directories

- `/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`).

Windows MySQL Server Installation Directory



ORACLE

Copyright © 2019, 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.

Bin 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”)

Data 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) and the `test` subdirectory for the `test` database (can be used for testing purposes).

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



Copyright © 2019, 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 © 2019, 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.”

Server Helper Programs

- **mysql.server:**
 - Used as a wrapper around `mysqld_safe` for systems such as Linux and Oracle Solaris that are using System V run-level directories
- **mysqld_safe:**
 - It sets up the error log and then launches `mysqld` and monitors it.
 - If `mysqld` terminates abnormally, `mysqld_safe` restarts it.
 - If the server does not start properly, look in the error log.
- **mysqld_multi:**
 - Perl script that simplifies the management of multiple servers on a single host. It can start or stop servers and report whether servers are running.



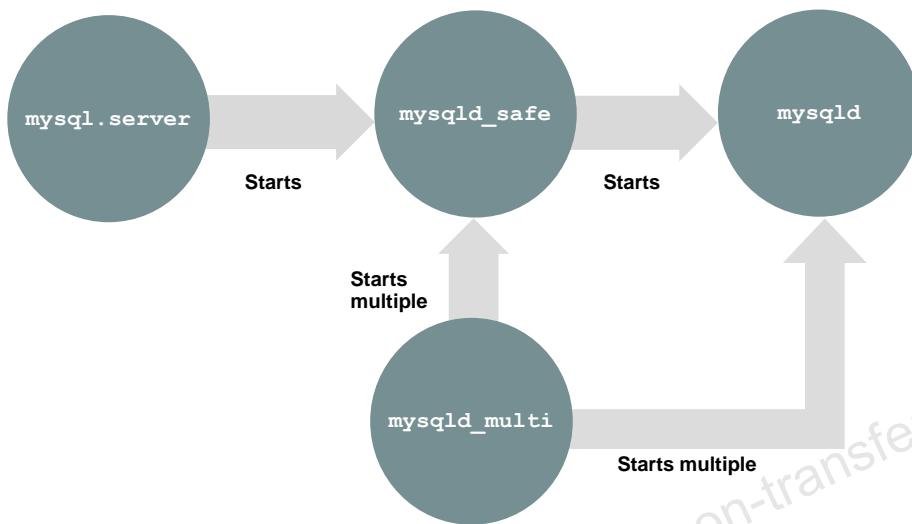
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Install the correct script to have the server run automatically at startup:

- On BSD-style Linux systems, it is most common to invoke `mysqld_safe` from one of the system startup scripts, such as the `rc.local` script in the `/etc` directory.
- Linux and UNIX System V variants with run-level directories under `/etc/init.d` use the `mysql.server` script. Prebuilt Linux binary packages install `mysql.server` under the name `mysql` for the appropriate run levels. Invoke it manually with an argument of `start` or `stop`, either directly or by using the `service` command.

Note: Starting multiple MySQL servers with `mysqld_multi` is covered in the lesson titled “Configuring MySQL.”

How the Server Helper Programs Interact



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

There are several ways to start the MySQL server process:

- Launch `mysqld` directly at the command line.
- Launch the `mysqld_safe` script, which in turn launches `mysqld`.
- Launch the `mysql.server` script by using the SysV Init process.
 - **Example:** Link `mysql.server` or copy it to `/etc/init.d/mysql` and run `service mysql start`.
- Launch `mysqld_multi` with a configuration that launches `mysqld_safe` (or `mysqld`) multiple times.

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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

mysql_secure_installation

- Invoke the security program:

```
$ mysql_secure_installation
```

- Prompts for actions to perform
- Applicable to UNIX and Linux operating systems only

- Security improvements:

- Sets or changes password for root accounts
- Removes any remote root accounts
- Removes any anonymous accounts
- Removes the test database if it exists



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Initial Passwords

When you install MySQL from an RPM package, the installation process sets up a random password for the `root` account and writes it to the server's log file (`/var/log/mysql/mysqld.log`) by default. When you install MySQL from a DEB package, the `dpkg` installer, by default, presents an interface in which you select the initial `root` password. However, if you have selected a noninteractive `dpkg` front end, this interface does not appear, and the initial `root` password remains blank.

For some other installations, such as installations from binary archive or from source, the initial passwords are blank. As a result, you should set up passwords as soon as the server has been started. In such situations, invoke `mysql_secure_installation` without arguments, which prompts you to determine which actions to perform.

This utility is also useful when you work with previous versions of MySQL that created multiple `root` users and created a test database with relaxed permissions. Builds of MySQL 8.0 provided by Oracle create only a single root account and do not create anonymous accounts or the test database. On such systems, you do not need to run the `mysql_secure_installation` utility if you have already changed the `root` password.

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



Copyright © 2019, 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
```



Copyright © 2019, 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 © 2019, 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.

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 © 2019, 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.  
...
```

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



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: a

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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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`



Copyright © 2019, 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 © 2019, 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



Copyright © 2019, 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 a new port mapping.
 - Set TCP port 3307 to the `mysqld_port_t` type so that MySQL can use it:

```
# semanage port -a -t mysqld_port_t -p tcp 3307
```
- 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 fcontext -a -t mysqld_db_t "/datadir(/.*)?"# restorecon -Rv /datadirrestorecon reset /datadir contextunconfined_u:object_r:default_t:s0->unconfined_u:object_r:mysqld_db_t:s0
```



Copyright © 2019, 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 © 2019, 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.
 - Source and archive installations create blank root passwords.
- 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';
```

ORACLE

Copyright © 2019, 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 © 2019, 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.

Topics

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



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Starting and Stopping MySQL

- Methods of starting MySQL:
 - Run the `mysqld` binary directly.
 - Run `mysqld_safe`.
 - Run `mysql.server` with a parameter.
 - Includes copies such as `/etc/init.d/mysql`
 - Example (upstart/SysV init): `service mysql start`
 - Example (systemd): `systemctl start mysql`
- Methods of stopping MySQL:
 - Kill the `mysqld` binary with the **SIGTERM signal (-15)**.
 - Kill `mysqld_safe` first if it is running.
 - Run `mysql.server` with the parameter value `stop`:
 - `service mysql stop`
 - `systemctl stop mysql`
 - `mysqladmin shutdown`



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Using `kill`

The `service mysql stop` command (or `systemctl stop mysql` 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` shuts down immediately without an orderly shutdown, as if you had pulled the power cord on the machine. If the `mysqld_safe` helper program is running, it detects this sudden shutdown as a crash and restarts `mysqld`.

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 3307 shutdown
```

```
Enter password: password
```



Copyright © 2019, 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/mysql`.
 - Some package installers create this file automatically.
 - Call it with `start`, `stop`, or `restart` options.
 - Examples:

```
service mysql start
/etc/init.d/mysql 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 © 2019, 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
```



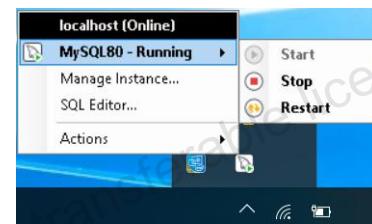
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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 © 2019, 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 © 2019, 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 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.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Downgrading

It is possible to downgrade between versions in the same series and from one major version to a previous major version, but downgrading is not as well tested as upgrading, nor is it as well supported. If you must downgrade, ensure that you have a strong rollback strategy in place.

Reading Release Notes

Before you upgrade, view the release notes for all versions between your existing version and the target version:

- For MySQL 8.0, use the following URL:
 - <http://dev.mysql.com/doc/relnotes/mysql/8.0/en/index.html>
- For previous versions, replace 8.0 in the URL.
 - Example: <http://dev.mysql.com/doc/relnotes/mysql/5.7/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 © 2019, Oracle and/or its affiliates. All rights reserved.

Searching for Specific Changes

If you know that an option or some other feature 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 8.0.15

```
mysqlsh -- util checkForServerUpgrade user@localhost:3306  
--target-version=8.0.15 --config-path=/etc/mysql/my.cnf
```



Copyright © 2019, 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."

Selecting an Upgrade Method

- **Physical (in-place):** Use this 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.
 - Run `mysql_upgrade`.
- **Logical (backup/restore):** Use this method when you upgrade to one or more later major versions.
 - Use `mysqldump` to back up the current databases.
 - Install a new empty MySQL server.
 - Restore the backed-up databases.
 - Run `mysql_upgrade`.



Copyright © 2019, 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-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.



Copyright © 2019, 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.

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.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: b, d

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

ORACLE®

Copyright © 2019, 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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Understanding MySQL Architecture

3



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide

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

ORACLE®

Copyright © 2019, 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
- Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin Interface

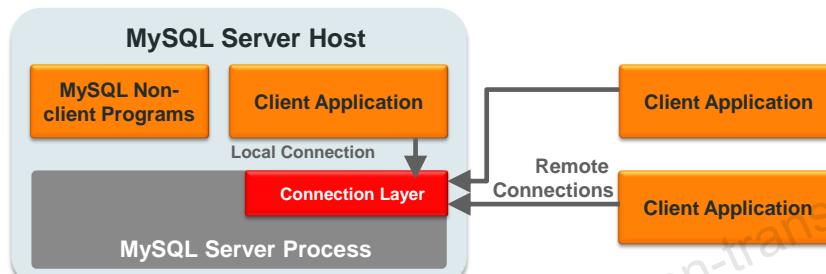


ORACLE®

Copyright © 2019, 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 © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

MySQL Server

- Is the database server program called mysqld
- Is not the same as a “host”
- Is 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 nontransactional tables
- Optimizes memory usage by using caches and buffers



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

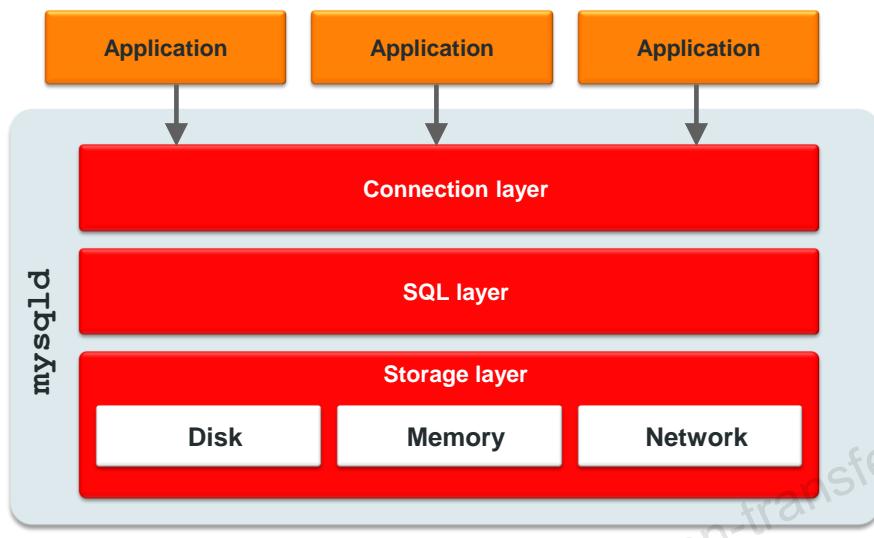
Terminology: Server and Host

- **Server:** A software program (`mysqld`) with a version number and a list of features
- **Host:** The physical machine on which the server program runs, which includes the following:
 - Its hardware configuration
 - The operating system running on the machine
 - Its network addresses
- Multiple servers can run on a single host.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Server Process



ORACLE®

Copyright © 2019, 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

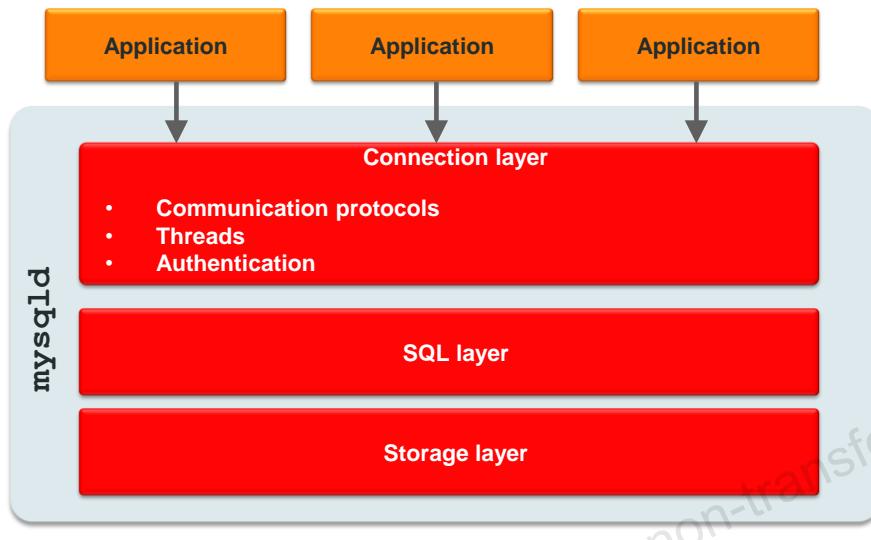
- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- How MySQL Stores Data
- How MySQL Stores Metadata
- Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin Interface



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Connection Layer



ORACLE®

Copyright © 2019, 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 communication protocols:

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

Communication Protocols

- Protocols are implemented in the client libraries and drivers.
- The speed of a connection protocol varies with the local settings.

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



Copyright © 2019, 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.

Local and Remote Communications Protocol: TCP/IP

- TCP/IP (Transmission Control Protocol/Internet Protocol):
 - Is the suite of communication 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
 - Enables connections between different hosts
- Example using a host name and the default port:

```
mysql --host=mysqlhost1 -uroot -p
```
- Example using an IP address and an alternative port:

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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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 Communication 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 (-S)` option
 - The client specifies the socket file 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`.



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 © 2019, 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 Communications Protocols in Windows: Shared Memory and Named Pipes

- Shared memory:
 - The server creates a named shared memory block that client processes on the same host can access.
 - Shared memory is disabled by default.
 - Example 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 © 2019, Oracle and/or its affiliates. All rights reserved.

Disabled by Default

Both shared memory and named pipe communications 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.

SSL by Default

The connection layer uses secure, encrypted connections where 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.



Copyright © 2019, 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 allows multiple 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.



Copyright © 2019, 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

Topics

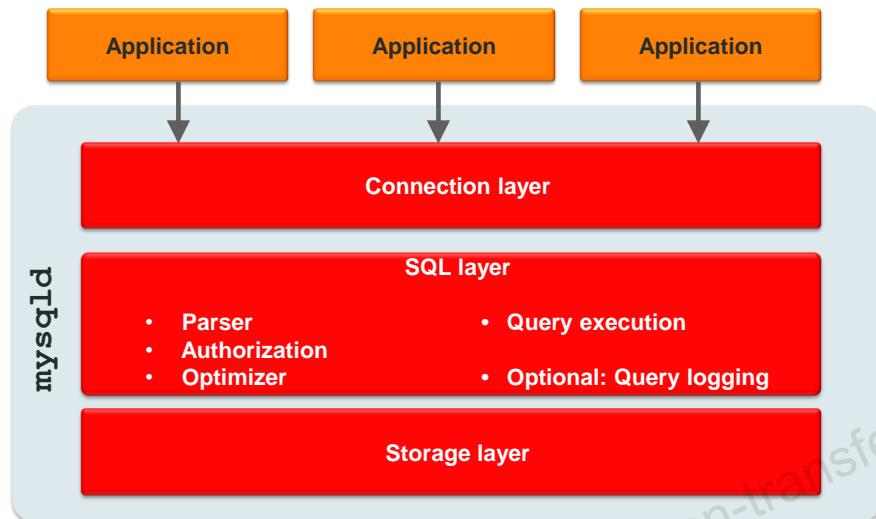
- Architectural Overview
- How MySQL Transmits Data
- **How MySQL Processes Requests**
- How MySQL Stores Data
- How MySQL Stores Metadata
- Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin Interface



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

SQL Layer



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

SQL Layer Components

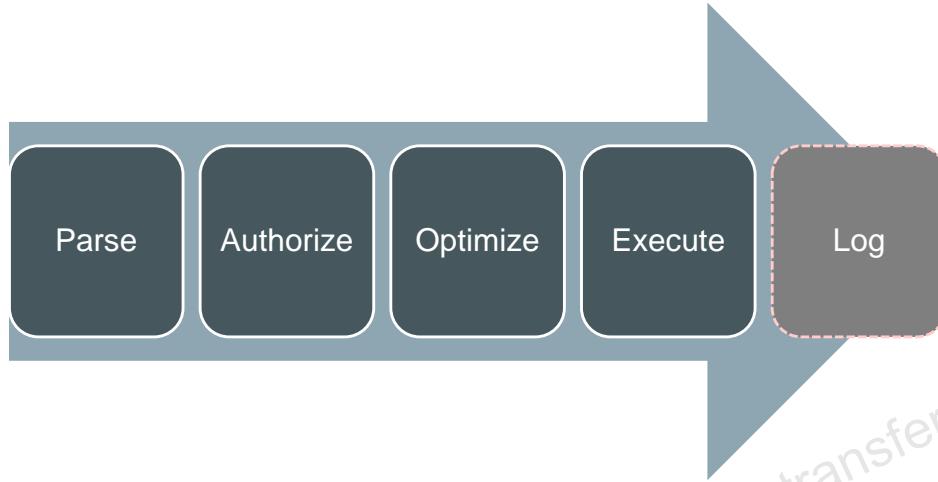
After a connection is established, MySQL processes each query in the SQL layer, which comprises the following components:

- **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:** Fulfils the execution plan for each query
- **Query logging:** Logs queries that the server receives or executes



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

SQL Statement Processing



ORACLE®

Copyright © 2019, 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.

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



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: d

Topics

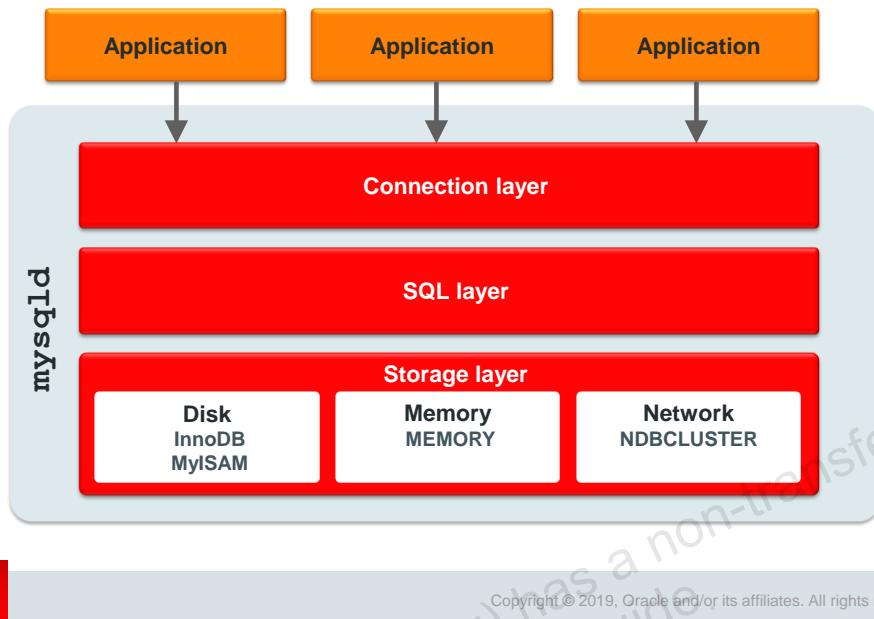
- Architectural Overview
- How MySQL Transmits Data
- How MySQL Processes Requests
- **How MySQL Stores Data**
- How MySQL Stores Metadata
- Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin Interface



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Storage Layer



ORACLE®

Copyright © 2019, 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.



Copyright © 2019, 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 by sending it to 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 © 2019, 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:
 - Only InnoDB and NDB support:
 - Foreign keys
 - Transaction control operations, such as `COMMIT` and `ROLLBACK`
 - Only InnoDB and MyISAM support full-text indexes.



Copyright © 2019, 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)



Copyright © 2019, 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.
 - File system (raw) backups are possible with simpler engines.
 - Example: MyISAM has 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 © 2019, Oracle and/or its affiliates. All rights reserved.

InnoDB Features

Feature	Support	Feature	Support
B-tree indexes	Yes	Geospatial indexing support (R-tree)	Yes
Backup/point-in-time recovery [a]	Yes	Hash indexes	No [c]
Cluster database support	No	Index caches	Yes
Clustered indexes	Yes	Locking granularity	ROW
Compressed data	Yes	Multiversion Concurrency Control (MVCC)	Yes
Data caches	Yes	Replication support [a]	Yes
Encrypted data [b]	Yes	Storage limits	64TB
Foreign key support	Yes	T-tree indexes	No
Full-text search indexes	Yes	Transactions	Yes
Geospatial data type support	Yes	Update statistics for data dictionary	Yes



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

InnoDB is the default storage engine. It provides transactions, full-text indexing, and foreign key constraints and so is useful for a wide mix of queries. It is multipurpose and supports read-intensive, read/write, and transactional workloads.

Historically, InnoDB was available as a storage engine plugin, but in more recent versions of MySQL, it is built into the server. InnoDB cannot be disabled.

The table in the slide indicates whether InnoDB supports the listed features. Note the following qualifications:

- [a] Is implemented in the server, rather than in the storage engine
- [b] Is implemented in the server (via encryption functions), rather than in the storage engine
- [c] InnoDB utilizes hash indexes internally for its Adaptive Hash Index feature.

InnoDB Storage Engine: Highlights

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



Copyright © 2019, 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

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



Copyright © 2019, 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
- 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 © 2019, 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



Copyright © 2019, 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.

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

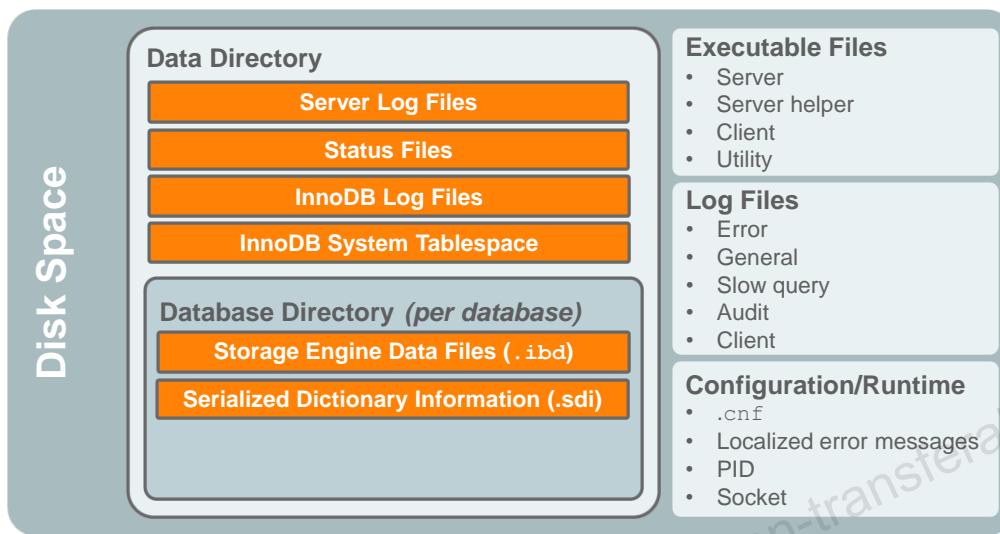


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Note

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

How MySQL Uses Disk Space



ORACLE

Copyright © 2019, 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 Files

- 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 log files for all databases reside at the data directory level.
 - InnoDB System Tablespace contains the data dictionary, undo log, and buffers.
- 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 © 2019, 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.

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**
- Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin Interface



ORACLE®

Copyright © 2019, 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:

- Column definitions
- Index definitions
- Constraints

ID	Name	Department	Salary
1	Sarah Finch	Sales	100000

Metadata

Data

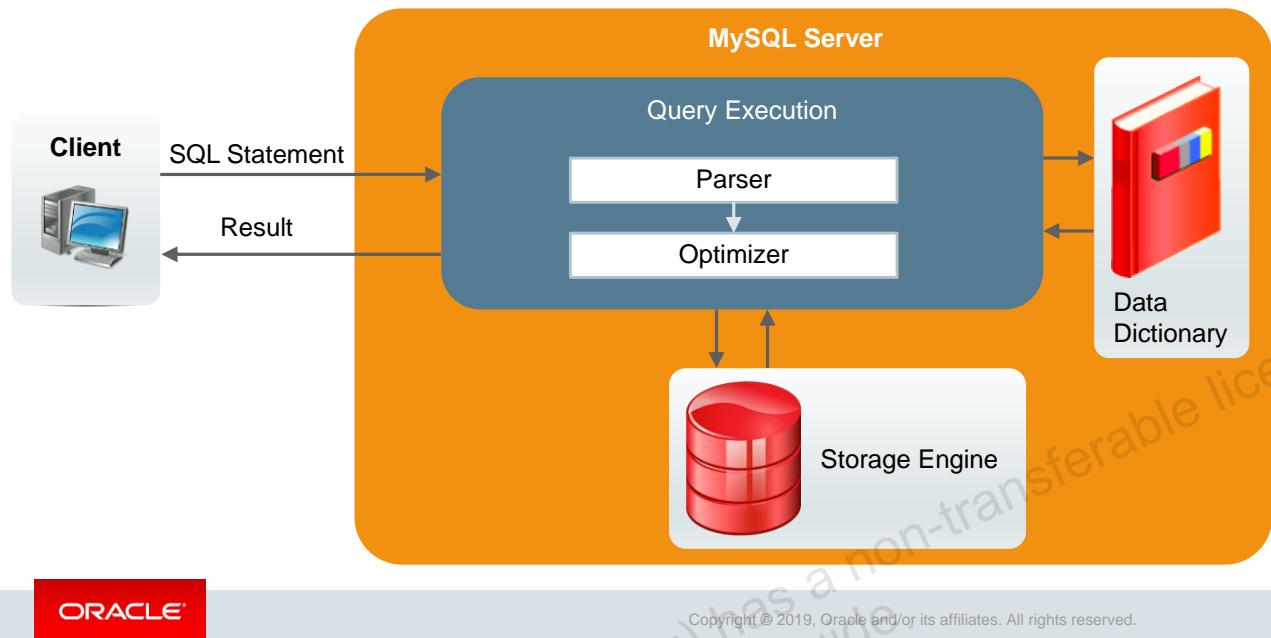
- The data dictionary is a centralized repository of metadata for all the data in an RDBMS.



Copyright © 2019, 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.

Role of the Data Dictionary

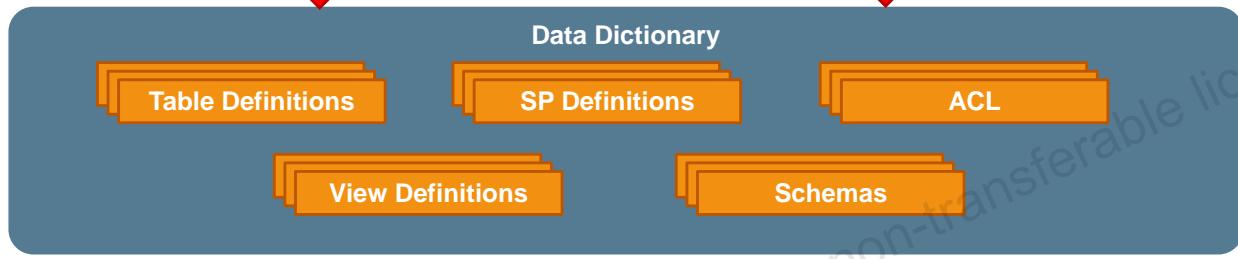


When a client executes a query, it communicates with the query executor on the server, which in turn communicates with the data dictionary and also possibly the storage engine to retrieve the data and return the results.

Types of Metadata

```
CREATE TABLE employees(  
    ID INT AUTO INCREMENT  
    ...  
    PRIMARY KEY (id),  
    INDEX ...  
    FOREIGN KEY ...  
)
```

```
CREATE PROCEDURE myProc(i  
    INT)  
SQL SECURITY INVOKER  
BEGIN  
    ...  
END
```

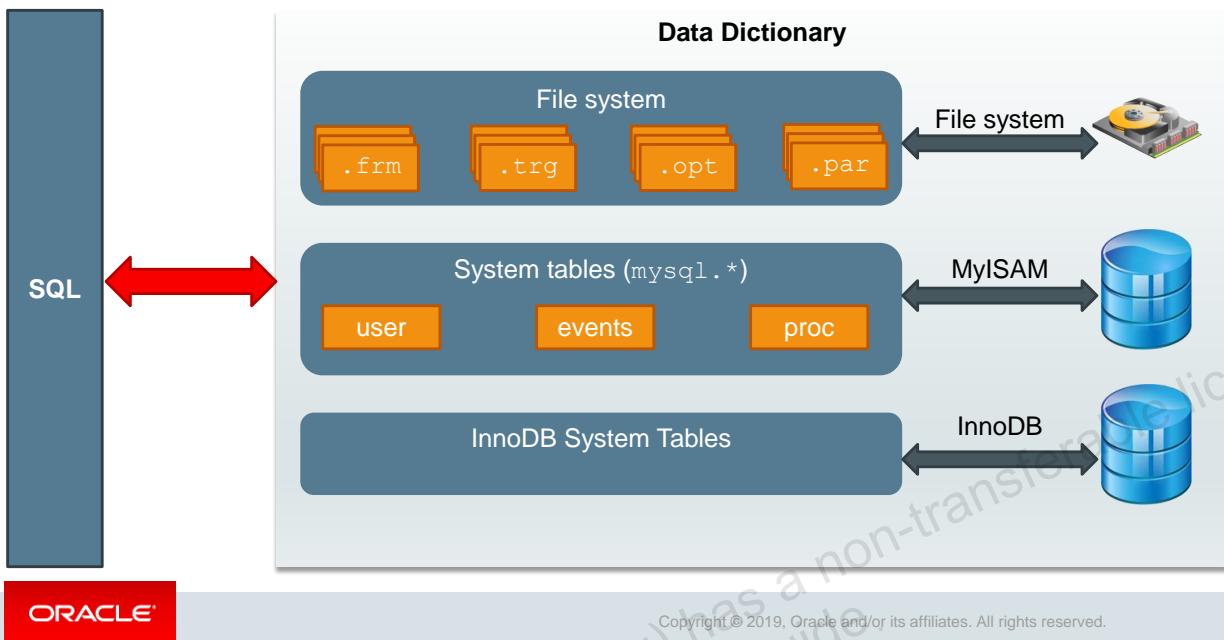


ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

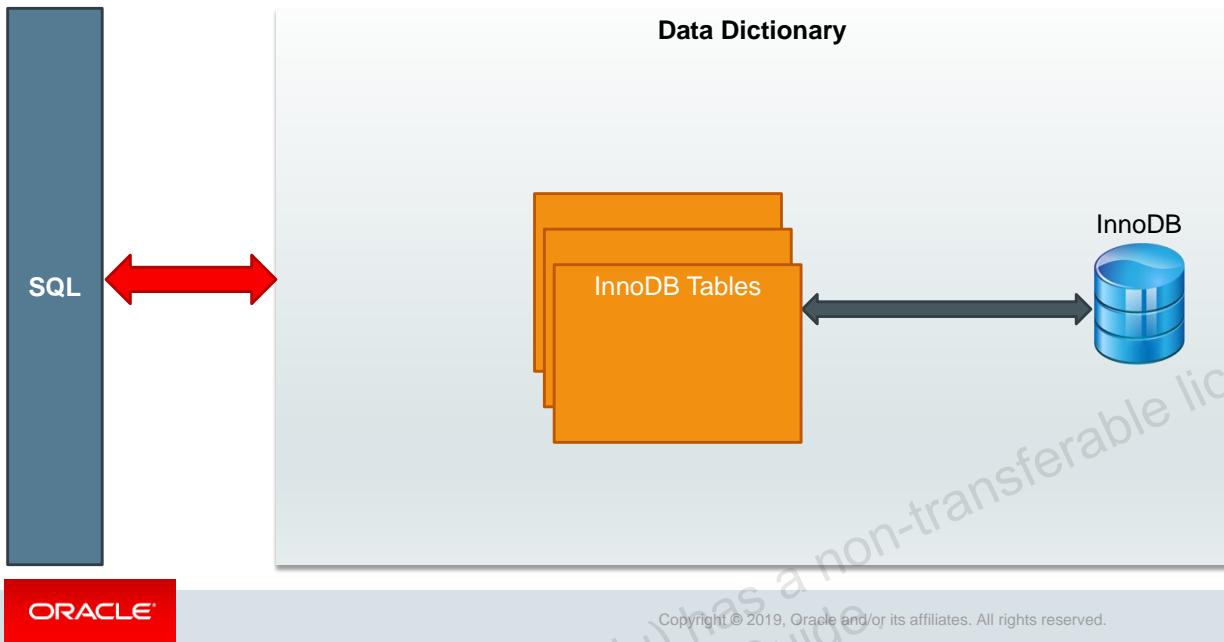
When you create a database object, its metadata is stored in the data dictionary. For example, if you create a table, then information about its columns, indexes, and so on is stored. If you create a stored procedure, then the data dictionary stores metadata such as invokers and privileges in the Access 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 file system. Events, stored procedures, the user tables, and access control information were stored in nontransactional 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 nontransactional 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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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 © 2019, 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.

Topics

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



ORACLE®

Copyright © 2019, 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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

InnoDB System Tablespace

- InnoDB stores metadata and buffers in the system tablespace, which consists of:
 - InnoDB Data Dictionary: Table, index, and column metadata
 - Change buffer: Changes to secondary index pages
 - Doublewrite buffer: Ensures crash-safe writes
- This is a single logical storage area, comprising one or more files in the data directory.
 - Typically, ibdata1, ibdata2, ...
- Each file can be a regular file or a raw partition.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Options That Control Tablespaces

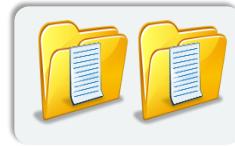
- The `innodb_data_file_path` option configures the size and physical location of the InnoDB system tablespace files on disk.
 - Default value: `ibdata1:12M:autoextend`
 - One file called `ibdata1`, 12 MB in size, autoextending
 - Example value: `ibdata1:20M;/ext/ibdata2:10M:autoextend`
 - Two files:
 - `ibdata1` in the data directory, fixed at 20 MB in size
 - `ibdata2` in the `/ext` directory, 10 MB in size, autoextending
 - If you have a set of files in the system tablespace, only the last file in the set can be autoextending.
 - Previous files in the set are fixed in size.
 - If the last file fills up, InnoDB automatically increases its size.
- 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 © 2019, Oracle and/or its affiliates. All rights reserved.

File-per-Table Tablespaces

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



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

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

- Example using an explicit TABLESPACE clause:

```
CREATE TABLE ftp_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 © 2019, 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;
```



ORACLE®

Copyright © 2019, 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 file system.
- General tablespaces provide the following benefits:
 - Less file system 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 © 2019, 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 file system target.
 - Example:

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



Temporary Tablespaces

- Temporary tables are created:
 - By MySQL to perform some operations during complex queries
 - Internal tables use the MEMORY storage engine.
 - You have no direct control over when it does this.
 - By users to store user data for use within a session
- 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 autoextending data file called `ibtmp1` in the `innodb_data_home_dir` directory.
- For information about the InnoDB temporary tablespace:

```
mysql> SELECT FILE_NAME, TABLESPACE_NAME, ENGINE,
-> INITIAL_SIZE, TOTAL_EXTENTS*EXTENT_SIZE
-> AS TotalSizeBytes, DATA_FREE, MAXIMUM_SIZE
-> FROM INFORMATION_SCHEMA.FILES
-> WHERE TABLESPACE_NAME = 'innodb_temporary'\G
```

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Metadata of temporary tables is available from
`INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO`.

Topics

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



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Redo Logs

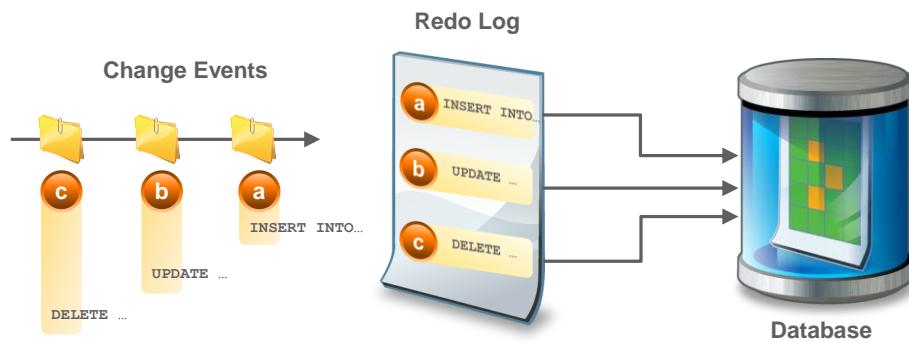
- Store InnoDB change operations before they are made to the data files
 - Enables InnoDB to optimize data writes so that they do not need to occur synchronously
- Are used during crash recovery
 - InnoDB replays operations in the redo log files to ensure transactional consistency across all tables, even for operations that did not write to the data files before the crash.
- Are located in the data directory by default
 - Typically, `ib_logfile0` and `ib_logfile1`
 - Controlled by the `innodb_log_group_home_dir` option



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The `innodb_log_files_in_group` option controls how many log files InnoDB uses to contain the redo log. Its default and recommended value is 2.

Redo Logs



ORACLE®

Copyright © 2019, 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

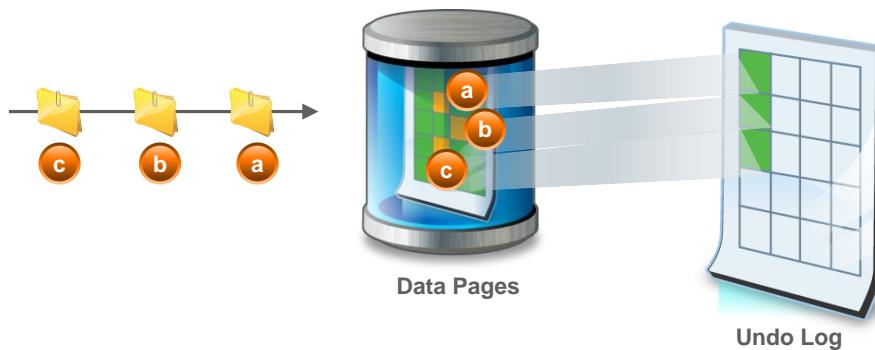
- Store copies of unmodified data that becomes modified by transactions so that InnoDB can access an earlier version of the data
- Are also called *rollback segments*
- Are stored by default in the *undo tablespace*
- Are used for MVCC and rollback
 - InnoDB retrieves the unmodified data from the undo 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 © 2019, 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 © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

When you store undo logs outside the system tablespace, they appear in tablespace files in the undo directory named `undoN`, where N is an integer (including leading zeros).

Temporary Table Undo Log

Undo logs for temporary tables are handled differently from other tables.

- They do not require redo logs.
 - After a crash, InnoDB does not rebuild temporary tables.
- They are stored in dedicated rollback segments in the temporary tablespace file.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Temporary table undo logs reside in the temporary tablespace and are used for 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 involves 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



ORACLE®

Copyright © 2019, 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
- Tablespaces
- Redo and Undo Logs
- **How MySQL Uses Memory**
- Plugin Interface

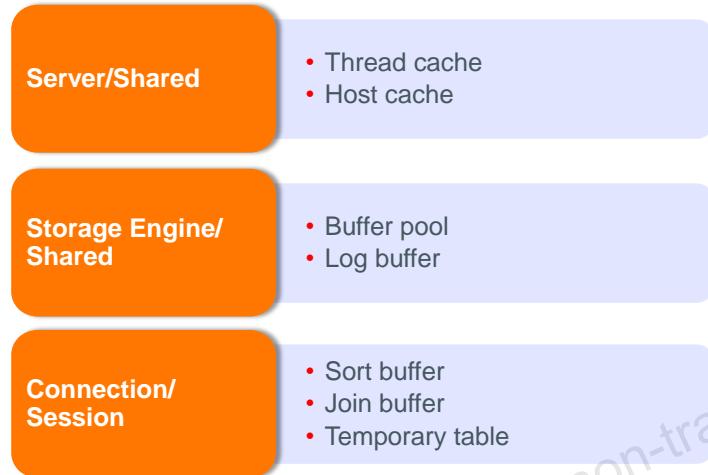


ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

How MySQL Uses Memory

- The MySQL server allocates memory in three different categories:



ORACLE

Copyright © 2019, 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 per session



Copyright © 2019, 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 buffers
 - Table open caches that hold descriptors for open tables



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

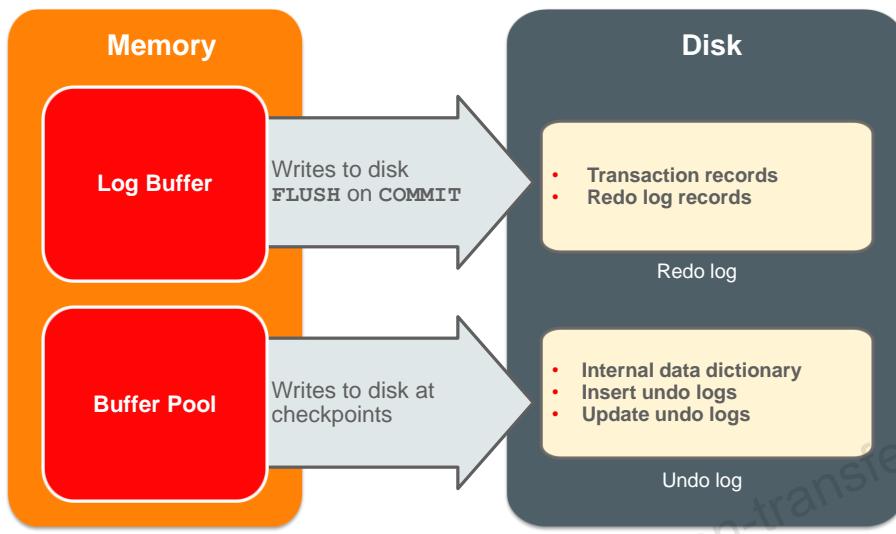
Session Memory

- 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 might use the 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 `join_buffer_size` of 1 MB with 100 connections means that there could be a total of 100 MB used for all join buffers simultaneously.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Log Files and Buffers



ORACLE®

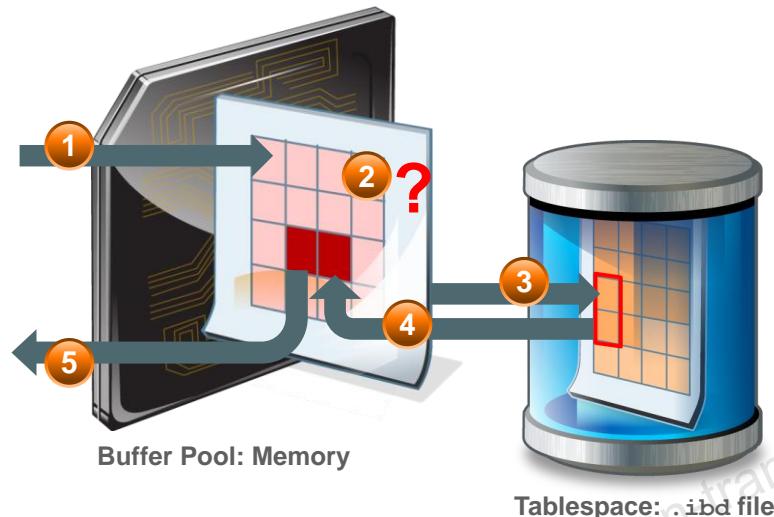
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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 reapplys the changes recorded in the logs to ensure that the tables reflect all committed transactions.

InnoDB Buffer Pool



ORACLE®

Copyright © 2019, 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 all remaining memory to the buffer pool
 - 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 © 2019, Oracle and/or its affiliates. All rights reserved.

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
- Tablespaces
- Redo and Undo Logs
- How MySQL Uses Memory
- Plugin Interface



ORACLE®

Copyright © 2019, 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 © 2019, 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.

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

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practices

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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

4

Configuring MySQL



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide

Objectives



After completing this lesson, you should be able to:

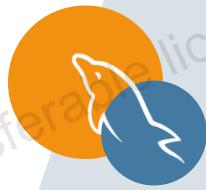
- Configure MySQL servers by using option files and command-line options
- Configure the `mysql` client
- Change MySQL settings dynamically
- Launch multiple MySQL servers on the same host

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Topics

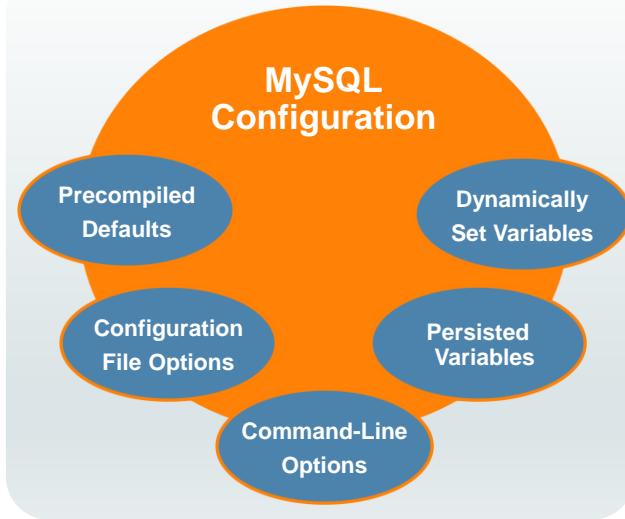
- Server Options, Variables, and the Command Line
 - Option Files
 - System Variables
 - Launching Multiple Servers on the Same Host



ORACLE®

Copyright © 2019, 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 which remain effective after the server restarts.

ORACLE

Copyright © 2019, 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
 - If you have multiple configuration files, options are applied from them in order, with later configuration files overriding options in earlier files.
- Configuration file options
- 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



Copyright © 2019, 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
```

- Unlike other variants of the mysqld command, this command does not start the mysqld process.



Copyright © 2019, 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 © 2019, 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 © 2019, 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.

Topics

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



ORACLE®

Copyright © 2019, 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 © 2019, 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.
- There is no single **my.cnf** or **my.ini** file.
 - The server might read multiple **my.cnf** or **my.ini** files from different locations.



Copyright © 2019, 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.

```
mysqld --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 © 2019, Oracle and/or its affiliates. All rights reserved.

The following command filters the output of the command:

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

If you launch MySQL from a script or service helper instead of executing the `mysqld` 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 © 2019, 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_safe]`
- `...`



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 2> /dev/null | 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 © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

Client Options: Examples

- **user** and **password**: Plain text authentication credentials
 - Use `mysql_config_editor` to create encrypted credentials.
 - Creates a `.mylogin.cnf` file that contains the credentials
- **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 © 2019, 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.
- 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 © 2019, 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='\u0@h[\d]> '
```



Copyright © 2019, 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
- 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 © 2019, 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.
- Example:

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

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



Copyright © 2019, 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 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 © 2019, 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 © 2019, 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]



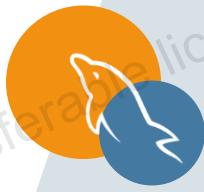
ORACLE®

Copyright © 2019, 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 © 2019, 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.



Copyright © 2019, 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`, `max_join_size`
 - Session only: `timestamp`, `error_count`



Copyright © 2019, 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
```

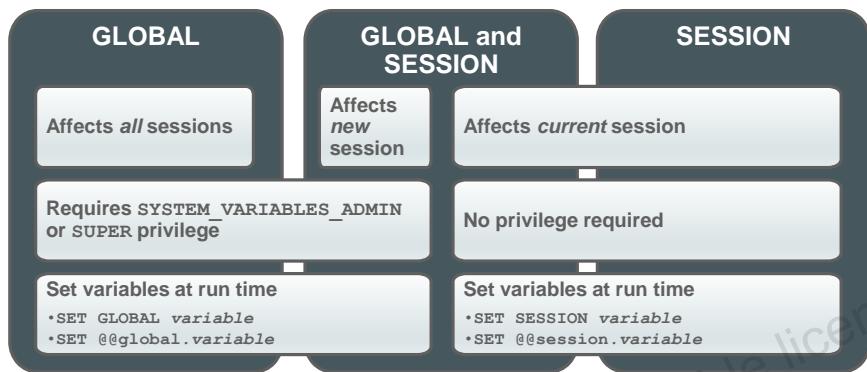


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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.

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 © 2019, Oracle and/or its affiliates. All rights reserved.

System Variable Types

- Use a string value for variables that have a string type such as CHAR or VARCHAR.
- Use a numeric value for variables that have a numeric type such as INT or DECIMAL.
- Set variables that have a Boolean (BOOL or BOOLEAN) type to 0, 1, ON, or OFF. (If they are set on the command line or in an option file, use the numeric values.)
- Set variables of an enumerated type to one of the available values for the variable.
 - You can also set them to the number that corresponds to the desired enumeration value.
 - For enumerated server variables, the first enumeration value corresponds to 0. This differs from ENUM columns, for which the first enumeration value corresponds to 1.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

To view the data type of any system variable, see: <http://dev.mysql.com/doc/mysql/en/server-system-variables.html>.

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 © 2019, 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** and **session_variables** tables contain global variables and current session variables.
- The **variables_by_thread** table contains session variables for all active threads.
 - Referencing the **thread_id**, detailed in the **threads** table

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

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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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 © 2019, 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.

Topics

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



ORACLE®

Copyright © 2019, 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 file system or network resources with other servers.



Copyright © 2019, 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_safe --socket=/mysql/socket2 --port=3312 --datadir=/mysql/data2  
mysqld_safe --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
```

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 © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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 complete doing so. All other connections stop logging them.
- d. New connections are prevented from executing the command `SET SESSION sql_notes=1`.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: b

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

ORACLE®

Copyright © 2019, 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`



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

5

Monitoring MySQL



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide

Objectives



After completing this lesson, you should be able to:

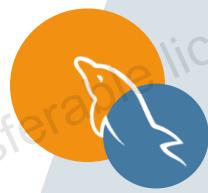
- Configure and view MySQL log files
- Identify slow queries
- Configure MySQL Enterprise Audit
- Use MySQL variables to monitor activity in MySQL
- Use MySQL Enterprise Monitor to view activity in MySQL

ORACLE®

Copyright © 2019, 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 © 2019, 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
- **General query log:** All statements that the server receives from clients
- **Slow query log:** Queries that take a long time to execute
- **Audit log:** Policy-based audit for Enterprise Edition
- **Binary log:** Data modifications



Copyright © 2019, 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
- Are written in text format
 - Except binary log



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Log File Usage Matrix

Log File	Server Options	File Name	Table Name	Programs
Error	--log-error --log-syslog	<i>host_name.err</i> Output to syslog		N/A
General	--general_log	<i>host_name.log</i>	general_log	N/A
Slow Query	--slow_query_log --long_query_time	<i>host_name-slow.log</i>	slow_log	mysqldumpslow
Binary	--log-bin --expire-logs-days	binlog.000001		mysqlbinlog
Audit	--audit_log --audit_log_file ...	audit.log		N/A



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Enabling Logs

- **Error log:** Set `--log-error=filename` to log errors to the given file.
 - Alternatively, use the `--log-syslog` server option to send error messages to `syslog` on UNIX or UNIX-based systems.
- **General query log:** Set `--general-log-file=filename` to log queries.
 - The global `general_log` and `general_log_file` server variables provide runtime control of the general query log.
 - Set `general_log` to 0 (or OFF) to disable the log or to 1 (or ON) to enable it.
- **Slow query log:** Set `--slow-query-log-file=filename` to provide runtime control over the slow query log.
 - Set `slow_query_log` to 0 to disable the log or to 1 to enable it.
 - If a log file is already open, it is closed and the new file is opened.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If you set `--log-syslog` to ON on a UNIX or UNIX-based system, error output is directed to `syslog`. On Windows-based systems, the default behavior is for the MySQL server to write error information to the Windows Event Log within the Application log.

The following server options provide finer control over how information is sent to operating system logs for both UNIX-like and Windows-based systems:

- `--log-syslog-facility`: Which type of program is sending the message. The default is `daemon`.
- `--log-syslog-include-pid`: Whether to include the server process ID in each line of the output
- `--log-syslog-tag`: Enables you to customize how the server is identified in the error output

Enabling Logs

- **Binary log:** Enabled by default, use `--skip-log-bin` to disable binary logging.
 - The server uses the option value as a base name, adding an increasing sequential numeric suffix to the base name as it creates new log files.
 - These log files are stored in binary format rather than text format.
- **Audit log:** The audit log is provided as an Enterprise Edition plugin that, when loaded, can be used to log events into the file specified by the `audit_log_file` option.
 - Auditing constantly writes to the audit log until you remove the plugin or you turn off the auditing with the `--audit-log-policy=NONE` option setting.
 - Prevent the removal of the plugin by using `--audit-log=FORCE_PLUS_PERMANENT` as an option when the server is started.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If the `binlog_format` server option is set to `ROW`, the binary log records update operations as row changes rather than statements, so such operations do not appear in the general query log of the slave servers. If you set the option's value to `MIXED`, some update operations appear in the general query log of the slave servers, but some (those that the server cannot safely replicate as statements) do not appear.

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

```
1424 Query      /* mem dbpool.default */
update `mem_events`.events set lastUpdateTime=1397633565708 where id=7
1429 Query      /* mem dbpool.default */ update `mem_events`.events
set lastUpdateTime=date-and-time where id=4970
1424 Query      /* mem dbpool.default */ commit
1429 Query      /* mem dbpool.default */ commit
date  7:32:46  1434 Connect    root@localhost on mysql
          SET NAMES latin1
          SET character_set_results = NULL
```



Copyright © 2019, 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 viewed with the `mysqldumpslow` command-line program



Copyright © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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



Copyright © 2019, 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 © 2019, 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 © 2019, 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 recommente 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 general 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.



Copyright © 2019, 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 general and slow query logs 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 © 2019, 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

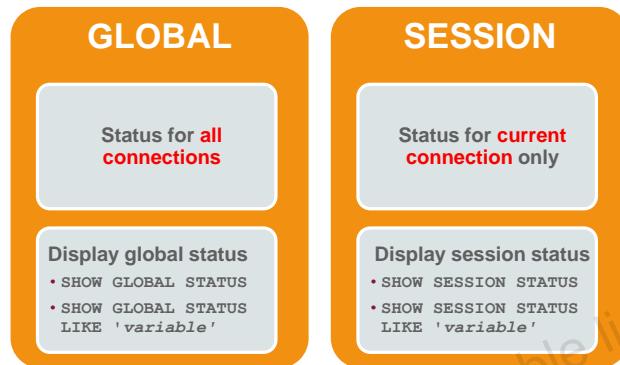


ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Status Variables

- Use the **SHOW STATUS** statement to assess the health of a system.
- Use a scope modifier (**GLOBAL** or **SESSION**) to display only 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 © 2019, Oracle and/or its affiliates. All rights reserved.

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     |
...
```

ORACLE

Copyright © 2019, 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 © 2019, 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
```

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



ORACLE

Copyright © 2019, 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 © 2019, 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.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Performance Schema Table Groups

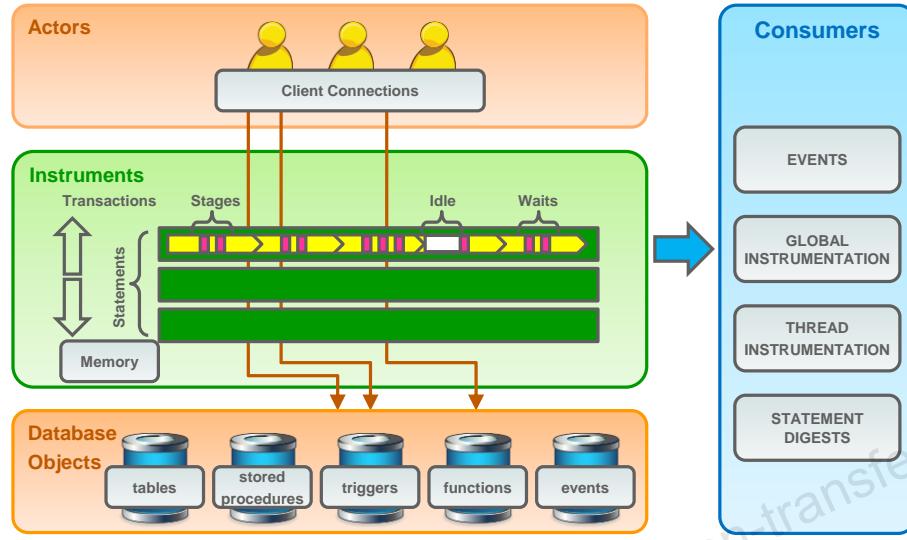


ORACLE®

Copyright © 2019, 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 © 2019, 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



Copyright © 2019, 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 © 2019, 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 © 2019, 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: 201660494489586
    TIMER_END: 201660494576112
    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 © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

Configuring MySQL Enterprise Audit

- Implemented using the `audit_log` server plugin together with other components
 - Available with Enterprise Edition subscriptions
- Policy-based logging:
 - Is set with the `audit_log_policy` option
 - Offers the choice of logging ALL, NONE, LOGINS, or QUERIES
 - Defaults to ALL
- Produces an audit record of server activity in a log file
 - Contents depend on policy 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



Copyright © 2019, 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 and user account data.
 - 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 © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

Audit Log File Contents

The audit log file is written as XML, using UTF-8.

- The root element is <AUDIT>.
- The closing </AUDIT> tag of the root element is written when the plugin terminates.
 - The tag is not present in the file while the audit log plugin is active.
- Each audit entry is an <AUDIT_RECORD /> element.

```
<?xml version="1.0" encoding="utf-8"?>
<AUDIT>
  <AUDIT_RECORD>
    ...
  </AUDIT_RECORD>
  <AUDIT_RECORD>
    ...
  </AUDIT_RECORD>
  ...
</AUDIT>
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Audit Records

The following is an example of an AUDIT_RECORD element:

```
<AUDIT_RECORD>
  <TIMESTAMP>2017-01-10T23:08:51 UTC</TIMESTAMP>
  <RECORD_ID>2_2017-01-10T23:08:46</RECORD_ID>
  <NAME>Connect</NAME>
  <CONNECTION_ID>1</CONNECTION_ID>
  <STATUS>0</STATUS>
  <STATUS_CODE>0</STATUS_CODE>
  <USER>root</USER>
  <OS_LOGIN></OS_LOGIN>
  <HOST>localhost</HOST>
  <IP></IP>
  <COMMAND_CLASS>connect</COMMAND_CLASS>
  <PRIV_USER>root</PRIV_USER>
  <PROXY_USER></PROXY_USER>
  <DB></DB>
</AUDIT_RECORD>
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Audit Log Attributes

- The `TIMESTAMP` of each audit record is in UTC.
- The `NAME` attribute 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` attribute provides the command status.
 - This is the same as the `Code` value displayed by the MySQL command `SHOW ERRORS`.
- Some attributes 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` attribute.



Copyright © 2019, 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 © 2019, 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



Copyright © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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



Copyright © 2019, 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 © 2019, 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

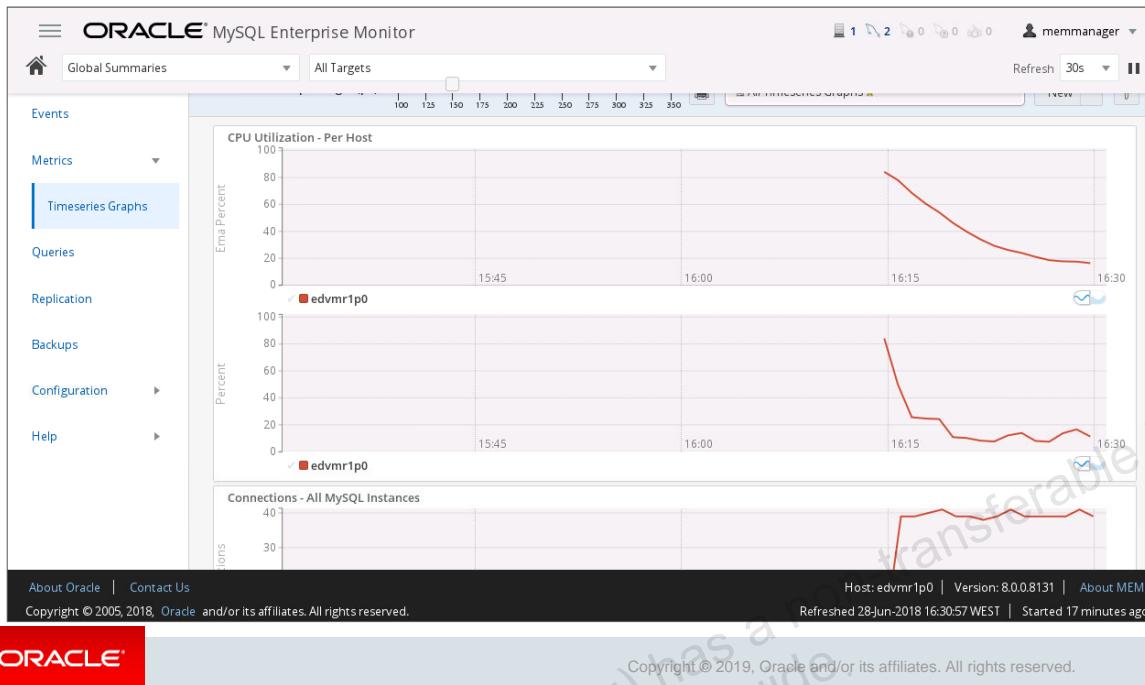
The screenshot shows the MySQL Instances dashboard. The main area displays a table of monitored MySQL instances:

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/enterprise...	
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/enterprise...	

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 with the 'Advisors' tab selected. The left sidebar lists categories: Overview, Events, Metrics, Queries, Replication, Backups, Configuration, Instances, Groups, and Advisors. The 'Advisors' category is highlighted with a blue background. The main pane displays 'Manage Advisors' with a table titled 'Administration'. The table shows various advisor categories with their counts: Administration (Count: 22), Agent (Count: 2), Availability (Count: 5), Backup (Count: 1), Graphing (Count: 81), and Memory Usage (Count: 6). The 'Memory Usage' row is highlighted with a red box labeled '1'. The first rule in the list, 'InnoDB Buffer Cache Has Sub-Optimal Hit Rate', is also highlighted with a red box labeled '2'. The 'Parameters' column for this rule is highlighted with a red box labeled '3', showing values 95, 85, and 75. The bottom status bar indicates the host is edvmrlp0, the version is 8.0.8131, and it was refreshed 28-Jun-2018 16:36:12 WEST, started 23 minutes ago.

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.

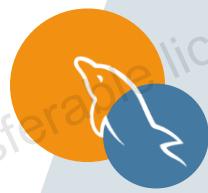
MySQL Enterprise Monitor: Events

The screenshot shows the MySQL Enterprise Monitor interface. The top navigation bar includes the Oracle logo, 'MySQL Enterprise Monitor', and user information ('memmanager'). The left sidebar has a tree view with nodes like 'Global Summaries', 'All Targets', 'Overview', 'Events' (selected), '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 sub-section 'Event Handlers' with a 'Create Event Handler' button. Below this is a table with columns: Handler Name, State, Subjects, Advisors, Statuses, and Actions. A search bar and pagination controls (Show 10 entries) are also present. The bottom of the screen shows footer links ('About Oracle', 'Contact Us'), copyright information ('Copyright © 2005, 2018, Oracle and/or its affiliates. All rights reserved.'), and system details ('Host: edvmrip0 | Version: 8.0.0.8131 | About MEM', 'Refreshed 28-Jun-2018 16:44:46 WEST | Started 32 minutes ago').

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.

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 © 2019, 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 © 2019, 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.

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 © 2019, 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.

Limits 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 © 2019, 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 © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

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



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: d

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

ORACLE®

Copyright © 2019, 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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

6

Managing MySQL Users



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide

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

ORACLE®

Copyright © 2019, 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 © 2019, 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.
 - 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.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Authentication and Authorization

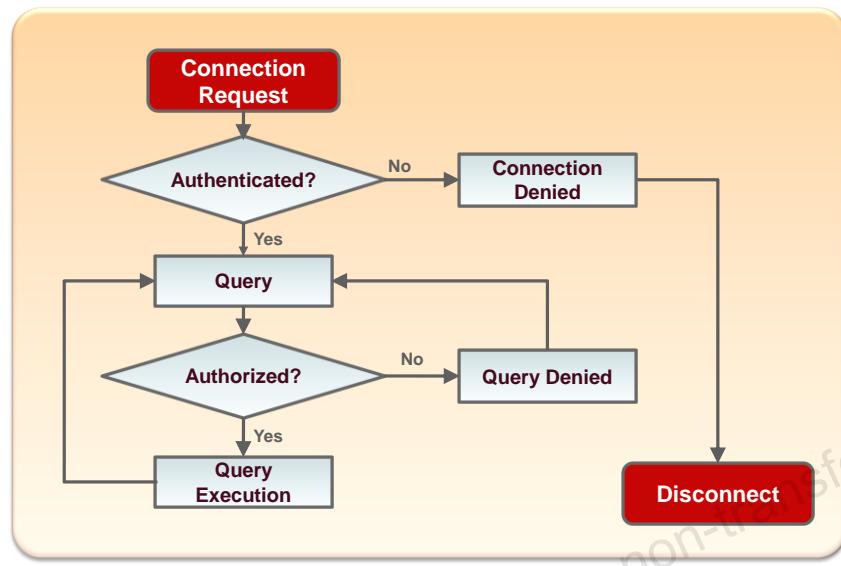
When you connect to a MySQL server and execute a query, it authenticates you and authorizes your activity.

- **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



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

User Connection and Query Process



ORACLE®

Copyright © 2019, 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;
+-----+-----+-----+
| user | host      | authentication_string          |
+-----+-----+-----+
| root | localhost | *2447D497B9A6A15F2776055CB2D1E9F86758182F |
+-----+-----+-----+
1 row in set (0.00 sec)
```

- 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 © 2019, 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.



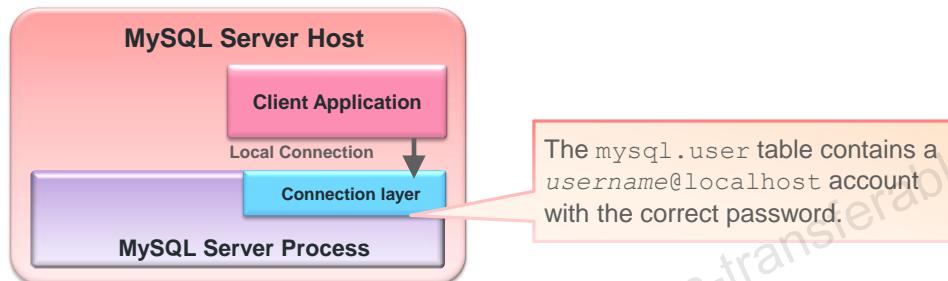
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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 -h localhost
```

- The default host name is `localhost`.
 - Indicates a local socket connection, not the DNS `localhost` synonym for `127.0.0.1`



ORACLE®

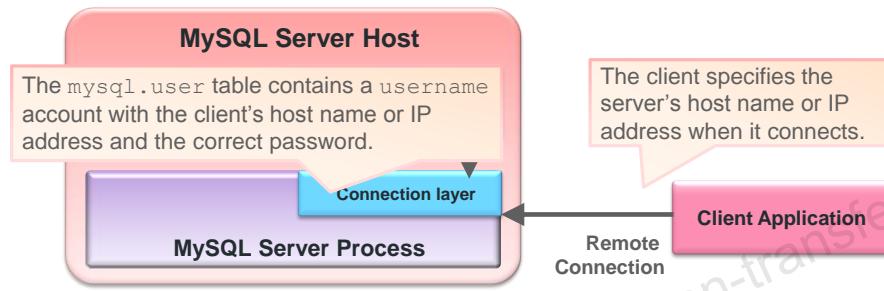
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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 to the server host.



ORACLE®

Copyright © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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 build 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 © 2019, 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 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.
 - But you can grant a user account to another user.



Copyright © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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*'

- Specify the username explicitly:

```
ALTER USER webuser@localhost IDENTIFIED BY 'n3W%P4$$w0rd';
```

- Change the password for the current user:

```
ALTER USER USER() IDENTIFIED BY 'n3W%P4$$w0rd';
```

- mysqladmin ... password '*newpassword*'

- Change the password for the user account you use to connect to the server:

```
mysqladmin -u webuser -p password 'n3W%P4$$w0rd';
```



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 must change their password before they can execute any other statements.



Copyright © 2019, 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 © 2019, 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 © 2019, 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`



ORACLE®

Copyright © 2019, 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 © 2019, 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
 - 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



Copyright © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

Some authentication methods, such as PAM (Pluggable Authentication Modules) 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.
- To load one of these plugins, start the server with the **plugin-load** option set to the plugin's file name.

```
[mysqld]
plugin-load=auth_test_plugin.so
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

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.

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 © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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 © 2019, 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.



Copyright © 2019, 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 © 2019, 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



Copyright © 2019, 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 file system.
- 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 © 2019, 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 © 2019, 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.



Copyright © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

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 © 2019, Oracle and/or its affiliates. All rights reserved.

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.



Copyright © 2019, 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 © 2019, 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 © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

User Privilege Restrictions

- You cannot explicitly deny access to a specific user on a specific object.
- 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.



Copyright © 2019, 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 that compare an access control list in a table with the output of the CURRENT_USER() function.

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.



Copyright © 2019, 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 the 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 © 2019, 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 © 2019, Oracle and/or its affiliates. All rights reserved.

Quiz

Q

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



ORACLE

Copyright © 2019, 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 © 2019, 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.



Copyright © 2019, 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.

```
SET PERSIST activate_all_roles_on_login = ON;
```



Copyright © 2019, 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 © 2019, 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();
```



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 © 2019, 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 © 2019, 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 © 2019, 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

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 © 2019, 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.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Before MySQL 8.0, grant tables used the MyISAM storage engine and were nontransactional.

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.
 - 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 © 2019, 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

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practices

- 6-1: Quiz – User Management
- 6-2: Creating Users and Roles
- 6-3: Granting Permissions



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

7

Securing MySQL

ORACLE®



MySQL™

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide

Objectives



After completing this lesson, you should be able to:

- Recognize common security risks
- List security problems and counter-measures for networks, passwords, operating systems, file systems, and applications
- Protect your data from interception and access
- Use SSL for secure MySQL server connections
- Use SSH to create a secure remote connection to MySQL
- Configure and use MySQL Enterprise Firewall

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Topics

- Security Risks
 - Network Security
 - Secure Connections
 - Password Security
 - Operating System Security
 - Protecting Against SQL Injections
 - MySQL Enterprise Firewall



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Security Risks

- MySQL server security is at risk when multiple users access it concurrently, particularly when those users connect over the Internet.
- It is not only the MySQL server that is at risk; the entire server host can be compromised.
- There are many types of security attacks:
 - Eavesdropping
 - Altering
 - Playback
 - Denial of service



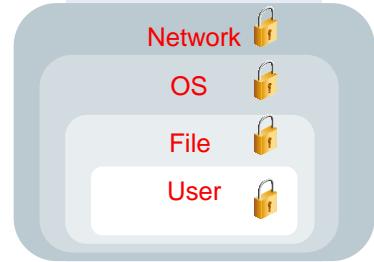
ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

MySQL Installation Security Risks

The following are the most common installation security risks:

- Network
 - MySQL server permits clients to connect over the network and make requests.
 - Network monitoring software can see client account information and data if passwords and network traffic are not encrypted.
 - Any account without a password is vulnerable to attack.
- Operating system
 - Extra user accounts on a server can increase the vulnerability of MySQL installation.
- File system
 - Directories, database files, and log files on the server can be opened by users who should not have access.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Network security can be thought of as the outermost level of defense. Within it is operating system security. Inside that is file system security, and even deeper is user security.

Topics

- Security Risks
- Network Security
- Secure Connections
- Password Security
- Operating System Security
- Protecting Against SQL Injections
- MySQL Enterprise Firewall



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

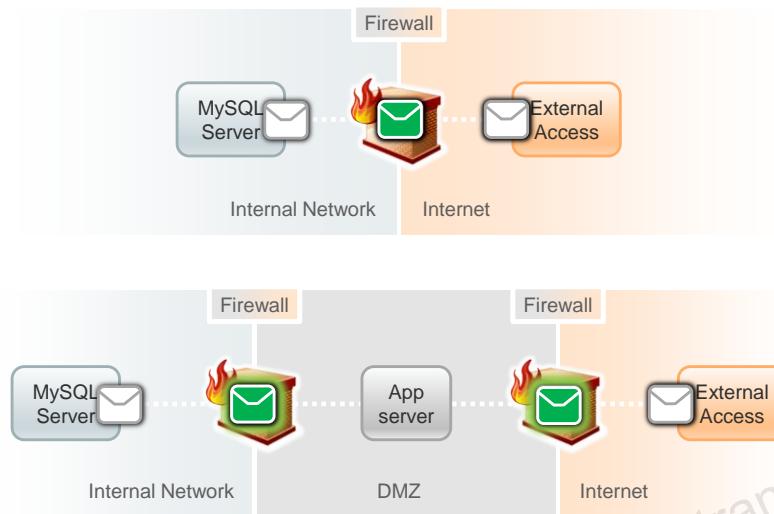
Securing MySQL from Public Networks

- MySQL uses the client/server model and provides an inherently network-oriented service.
 - Networked clients connect to the MySQL server.
- If MySQL is running on the same machine as an Internet-facing application that uses MySQL:
 - Disable networking.
 - Permit only socket connections.
- MySQL responds to requests on network ports from clients that can access those ports.
 - Do not connect MySQL server hosts directly to the Internet.
 - Use a firewall (or multiple firewalls arranged in a demilitarized zone [DMZ]) to prevent connections from unauthorized clients.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Using One or More Firewalls



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Animated Graphic (animation for each slide advance)

You can use one or more firewalls to prevent access to the MySQL server from unauthorized external access.

First advance

If you have a single firewall between the Internet and your internal network, unauthorized attempts can be rejected by the firewall rather than being passed on to your MySQL server.

Next advance

If external access is authorized by the firewall, it passes the request to the MySQL server.

Next advance

Responses from the MySQL server pass back out to the external access.

A DMZ is a network that is firewalled both from the Internet and from the company's internal network. For example, you can place an application server like a web server in the DMZ so that a firewall protects it from being accessed on any port that is not used by the application. That server might communicate through a second firewall to a MySQL server running in an internal network. In such an arrangement, MySQL is separated from the Internet by two firewalls.

Next advance

If access is accepted by the first firewall, it still might be rejected by the second firewall and not reach the MySQL server.

Next advance

Authorized accesses will pass through both firewalls and get to the MySQL server.

Next advance

The MySQL server responds to authorized requests.

Preventing Network Security Risks

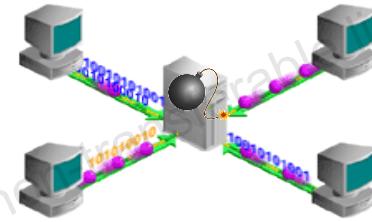
- Use a firewall to control access to MySQL ports.
- Consider limiting the network interfaces used by the server.
- Use a secure installation script to create a secure initial configuration:
`mysql_secure_installation`
- Ensure that MySQL accounts are protected with passwords and do not have unnecessary privileges.
- Do not grant more permissions than a user requires.
- Ensure that only authorized clients can connect to the server to access its databases.
- Do not transmit plain (unencrypted) data over networks.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Securing MySQL in Private Networks

- If the MySQL server port is visible to all users within a private network, it is subject to direct or indirect attacks.
 - direct attacks from disgruntled employees or snoopers
 - indirect attacks from external agents who use phishing attacks or Trojans to gain control of employee hosts
- Unencrypted data is accessible to users who have the ability to intercept it.
 - Network sniffing tools can view traffic on the same network segment.
 - Use an encrypted protocol such as SSL or SSH to encrypt traffic between clients and servers.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Topics

- Security Risks
- Network Security
- **Secure Connections**
 - Password Security
 - Operating System Security
 - Protecting Against SQL Injections
 - MySQL Enterprise Firewall



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Secure Connections

Transmissions over a network can be intercepted by a snooper and possibly modified.

Secure connections:

- encrypt transmissions and make any intercepted transmissions unreadable by the snooper
- use Secure Sockets Layer (SSL) and Transport Layer Security (TLS) based on the OpenSSL API
 - Many options and variables in MySQL refer to SSL, but it actually uses the updated and more robust TLS.
- include mechanisms to verify, identify, detect changes in transmissions, and prevent later replay
- can be enabled on a per-connection basis
 - It can be optional or mandatory for a given user account.
 - Some applications require the extra security afforded by secure connections.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

TLS uses encryption algorithms to ensure that data received over a public network can be trusted. It has mechanisms to detect data change, loss, or replay. TLS also incorporates algorithms that provide identity verification using the X509 standard.

X509 makes it possible to identify someone on the Internet. In basic terms, an entity called a “Certificate Authority” (or CA) assigns electronic certificates to anyone who needs them. Certificates rely on asymmetric encryption algorithms that have two encryption keys (a public key and a private key). A certificate owner can present the certificate to another party as proof of identity. A certificate consists of its owner's public key. Any data encrypted using this public key can be decrypted only using the corresponding private key, which is held by the owner of the certificate.

Both the Community and Enterprise editions include OpenSSL libraries and support TLSv1, TLSv1.1, and TLSv1.2, with TLSv1.2 being the most secure and preferred version.

The `Ssl_version` and `Ssl_cipher` status variables display which encryption protocol and cipher MySQL is using, respectively.

If you want to use wolfSSL instead of OpenSSL, you can compile the Community edition of MySQL from a source distribution with this option. This option is not available in the Enterprise edition.

Secure Connection: Overview

1. A client initiates a secure connection to a server.
2. The server provides a digital certificate to the client.
 - confirms the identity of the server
 - provides server's public key (*asymmetric encryption*)
3. Client uses server's digital certificate to verify server identity.
4. Client determines session key (*symmetric encryption*) and uses server's public key to encrypt transmission to the server.
5. Server uses its private key to decrypt the client's transmission.
 - Now only the client and server know the session key.
 - For the rest of the session, the client and server use the session key to encrypt and decrypt transmissions.
 - SSL includes mechanisms for detecting modifications and preventing replay.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

This is an overview of the process. There are other details that are part of the initial negotiation and ongoing transmissions. When the client initiates the secure connection, it sends the server the version of SSL/TLS to use, a cipher (encryption algorithm) to use for the session key, and a hash. The hash is used to generate a Message Authentication Code (MAC) that includes a hashed digest of the contents of each transmission and a message sequence number. These are included in each transmission to detect modifications and prevent replay. The symmetric encryption used for the session key is not as secure as the asymmetric encryption used to exchange the session key between client and server, but it is more efficient and less resource-intensive than asymmetric encryption.

Generating a Digital Certificate

- To support SSL, a server must have a digital certificate based on the X.509 standards, issued by a Certificate Authority (CA).
 - The CA verifies the identity of the server and provides the public key/private key pair for the asymmetric encryption.
 - CA can be a trusted third-party organization, or the server can act as its own CA and issue a self-signed digital certificate.
- MySQL 8.0 can act as a CA and generate a self-signed digital certificate.
 - When SSL is enabled, a server compiled with OpenSSL automatically checks for digital certificate files at startup and generates them if they are not present.
 - You can run `mysql_ssl_rsa_setup` if you want to generate digital certificate files with different options, such as locating the files in a different data directory.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

A Certificate Authority (CA) is a trusted third party that issues digital certificates. To receive a digital certificate from a trusted third party, a server administrator applies for the digital certificate, providing identifying information and possibly paying a fee. The CA signs the digital certificate and includes the public key. It also provides the private key for the server to keep secret. The CA also provides a copy of its own digital certificate to verify its identity. There are several well-known trusted third-party CAs that issue digital certificates.

If the server generates a self-signed digital certificate, the digital certificate for the CA will identify the server. A self-signed digital certificate is acceptable for testing or for connections from a client to a well-known server. If the client is accessing the server over the Internet, it is worth acquiring a digital certificate from a trusted third-party CA.

Server Security Defaults

- SSL is enabled by default on the server (–ssl=ON).
- Check whether a running server supports SSL by querying the value of the `have_ssl` system variable:

```
mysql> SHOW VARIABLES LIKE 'have_ssl';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| have_ssl     | YES   |
+-----+-----+
```

- Values returned:
 - YES: The server supports (but does not require) SSL connections and is ready to connect securely.
 - DISABLED: The server is capable of supporting secure connections, but secure connections were not enabled at startup.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The `have_openssl` system variable is an alias for `have_ssl`.

SSL Is Enabled by Default with MySQL Clients

- Client programs attempt to establish a secure connection by default for all TCP/IP connections.
- Check whether the current session is using SSL with the STATUS or \s command:

```
mysql> STATUS
-----
...
Current user:      root@localhost
SSL:              Cipher in use is DHE-RSA-AES128-GCM-SHA256
...
Connection:        localhost via TCP/IP
...
```

- For a client on the same host as the server, include the --protocol=TCP or --host=127.0.0.1 option. For example:

```
# mysql -u root -p --protocol=TCP
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If you are connecting a client to a server on the same host, MySQL uses a socket on UNIX or a named pipe on Windows. You do not need TCP/IP or a secure connection to connect a client and server on the same host. To force a client to connect over TCP/IP to test SSL, you can include --protocol=TCP or --host=127.0.0.1 in the mysql command-line client.

Disabling SSL on MySQL Server

1. Start MySQL server with either the `--ssl=0` or `--skip-ssl` option.
2. Log in to the server over TCP/IP and check whether SSL is enabled and if the connection is secure.

```
mysql> SHOW VARIABLES LIKE 'have_ssl';
+-----+-----+
| Variable_name | Value   |
+-----+-----+
| have_ssl     | DISABLED |
+-----+-----+
1 row in set (#.## sec)

mysql> STATUS
-----
...
SSL:          Not in use
...
Connection:   localhost via TCP/IP
...
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When SSL is enabled on the server, it supports secure connections but does not require the client to use them. You normally do not need to disable SSL on the server; it will still accept a connection from a client that is not using a secure connection.

You might need to disable SSL on the server for other reasons, such as testing. You can also use either the `--ssl=0` or `--skip-ssl` option. These options are not dynamic, so they can be issued only at startup. They can be included in a configuration file.

When the server is started with one of these options, the global server variable `have_ssl` is set to `DISABLED`. The connection status shows `SSL` as “Not in use.” If you restart the server with no `--ssl` option, the default is equivalent to including `--ssl=1` to enable SSL.

Setting Client Options for Secure Connections

Use the `--ssl-mode` option, which accepts the following values:

- **PREFERRED**: Establishes a secure connection if possible or falls back to an unsecure connection. This is the default if `--ssl-mode` is not specified.
- **DISABLED**: Establishes an insecure connection.
- **REQUIRED**: Establishes a secure connection if possible or fails if unable to establish a secure connection.
- **VERIFY_CA**: As for REQUIRED, but also verifies the server digital certificate with the Certificate Authority.
- **VERIFY_IDENTITY**: As for VERIFY_CA, but also verifies that the server digital certificate matches the MySQL server host.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Host Name Matching with `VERIFY_IDENTITY`

If the client uses OpenSSL 1.0.2 or higher, `VERIFY_IDENTITY` checks whether the host name it connects to matches either the Subject Alternative Name value or the Common Name value in the server certificate. Otherwise, the client checks whether the host name matches the Common Name value in the server certificate.

Client --ssl-mode Option: Example

With SSL enabled on the server, connect with SSL disabled for the client. Check the server and connection status.

```
# mysql -u root -p --protocol=TCP --ssl-mode=DISABLED
Enter password: *****

mysql> SHOW VARIABLES LIKE 'have_ssl';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| have_ssl      | YES   |
+-----+-----+
1 row in set (#.## sec)

mysql> STATUS
-----
...
SSL:          Not in use
...
Connection:    localhost via TCP/IP
...
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In this example, the server supports SSL, but the connection from the client does not use SSL. If, conversely, SSL is disabled on the server, and the client tries to connect with --ssl-mode=REQUIRED, the connection is rejected.

Setting the Permitted Versions for SSL/TLS for the Server

- Use the global `tls_version` server system variable.
 - By default, the value for `tls_version` is the list of all protocols supported by the SSL library MySQL is compiled against.
- Provide a comma-separated list of accepted versions.
 - Example, in a config file such as `/etc/my.cnf`

```
[mysqld]
tls_version=TLSv1.1,TLSv1.2
```
 - Reject connections via the less secure TLSv1 protocol.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Setting the Permitted Versions for SSL/TLS for the Client

- Use the `--tls-version` client system variable.
- Provide a comma-separated list of accepted versions.
 - Example:

```
# mysql --tls-version="TLSv1,TLSv1.1"
```

- Check the `Ssl_version` status variable for the version of TLS being used for the connection between client and server:

```
mysql> SHOW SESSION STATUS LIKE 'Ssl_version';
+-----+-----+
| Variable_name | Value   |
+-----+-----+
| Ssl_version  | TLSv1.1 |
+-----+-----+
1 row in set (#.## sec)
```

- The client and server establish the connection using the latest version of TLS that both support.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Setting the Cipher to Use for Secure Connections

A cipher specifies an encryption algorithm to use, including the length of the encryption key.

- More robust ciphers with longer keys are more secure.
- Ciphers have names like DHE-RSA-AES256-SHA or AES128-SHA.
- By default, the connection uses the most robust cipher supported by both the client and server.
- Clients and servers can use the **--ssl-cipher** option to specify a list of permissible ciphers, separated by colons.

– Example:

```
--ssl-cipher=DHE-RSA-AES256-SHA:AES128-SHA
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Global System Variable and Session Status Variables for Ciphers

The `--ssl-cipher` option sets a global system variable and two session status variables.

- **`ssl_cipher`**: Global server system variable with the list of permissible ciphers separated by colons. If the `--ssl-cipher` option is not set for the server, this is blank. Any supported cipher can be used and the most robust one available is selected.
- **`Ssl_cipher_list`**: Session status variable with the list of permissible ciphers separated by colons. If the `--ssl-cipher` option is not set for the server, this variable lists *all* available ciphers.
- **`Ssl_cipher`**: Session status variable that displays the cipher that is being used for the current session. For sessions that are not using a secure connection, this variable is blank.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If the `--ssl-cipher` option is set by the client, it affects only the cipher being used for that connection, that is, the `Ssl_cipher` session status variable, not the `ssl_cipher` global server system variable or the `Ssl_cipher_list` session status variable.

Cipher System and Status Variables: Example 1

The server is started with the following option:

```
--ssl-cipher=DHE-RSA-AES256-SHA:AES128-SHA
```

```
mysql> SHOW GLOBAL VARIABLES LIKE 'ssl_cipher';
+-----+-----+
| Variable_name | Value           |
+-----+-----+
| ssl_cipher    | DHE-RSA-AES256-SHA:AES128-SHA |
+-----+-----+
1 row in set (#.## sec)

mysql> SHOW SESSION STATUS like 'Ssl_cipher%';
+-----+-----+
| Variable_name | Value           |
+-----+-----+
| Ssl_cipher    | DHE-RSA-AES256-SHA
| Ssl_cipher_list | DHE-RSA-AES256-SHA:AES128-SHA |
+-----+-----+
2 rows in set (#.## sec)
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In this example, the values of the `ssl_cipher` server system variable and the `Ssl_cipher_list` session status variable are the same. The current connection is using the DHE-RSA-AES256-SHA cipher (the value of the `Ssl_cipher` status variable).

Cipher System and Status Variables: Example 2

The server is started without specifying --ssl-cipher.

```
mysql> SHOW GLOBAL VARIABLES LIKE 'ssl_cipher';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| ssl_cipher    |      |
+-----+-----+
1 row in set (#.## sec)

mysql> SHOW SESSION STATUS like 'Ssl_cipher%';
+-----+-----+
| Variable_name     | Value          |
+-----+-----+
| Ssl_cipher        | DHE-RSA-AES128-GCM-SHA256 |
| Ssl_cipher_list   | ECDHE-ECDSA-AES128-GCM-SHA256:
                           ECDHE-ECDSA-AES256-GCM-SHA384:
                           ECDHE-RSA-AES128-GCM-SHA256:...
+-----+-----+
2 rows in set (#.## sec)
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In this example, the values of the `ssl_cipher` server system variable and the `Ssl_cipher_list` session status variable are the same. The current connection is using the DHE-RSA-AES256-SHA cipher (the value of the `Ssl_cipher` status variable).

Setting Client SSL/TLS Options by User Account

Use the `REQUIRE` clause with a `CREATE USER` or `ALTER USER` statement with one of the following options:

- **NONE**: (Default) account has no SSL or X509 requirements and can use secure or non-secure connections.
- **SSL**: Account must use a secure connection.
- **X509**: Account must connect with a secure connection from a client that has a digital certificate for the client.
- **ISSUER 'issuer'**: Account must use a secure connection from a client with a certificate issued by the specified CA.
- **SUBJECT 'subject'**: Account must use a secure connection from a client that has a digital certificate with the specified Subject field identifying the owner of the certificate.
- **CIPHER 'cipher'**: Account must use a secure connection with the specified cipher.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Examples:

```
ALTER USER a@localhost REQUIRE SSL;
ALTER USER b@localhost REQUIRE X509;
ALTER USER c@localhost REQUIRE CIPHER 'DHE-RSA-AES256-SHA';
```

Generating a Digital Certificate

- When the MySQL server starts, or when `mysql_ssl_rsa_setup` executes, it checks for the following digital certificate files:
 - `ca.pem`: Digital certificate for the CA that issued the server's digital certificate
 - `server-cert.pem`: Digital certificate for the server verifying the server's identity and including the public key
 - `server-key.pem`: Private key for the server
- If those files are not present, it generates those files for a self-signed digital certificate and also creates the following files:
 - `ca-key.pem`: Private key for the CA
 - `client-cert.pem`: A client certificate to share with clients
 - `client-key.pem`: A client private key to share with clients



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If the server is acting as its own CA and generating a self-signed digital certificate, it needs a private key to go along with the public key in the digital certificate for the CA. The digital certificate for the CA is used if the client `--ssl-mode` is set to verify the CA. The digital certificate and private key for the client are used to support the X509, SUBJECT, and ISSUER options for the REQUIRE clause of the CREATE USER and ALTER USER statements.

The server distributes the client certificate file and private key file securely to client computers that require client digital certificates. If the server generates self-signed digital certificates, the issuer for all of them is `MySQL_Server_version_Auto_Generated_CA_Certificate`, where the version value is the version of the MySQL server. The Subject for the client digital certificate is `MySQL_Server_version_Auto_Generated_Client_Certificate`.

SSL Server Variables for Digital Certificates

- **ssl_ca**: The file that contains the list of trusted CAs. The default value is `ca.pem`. Change the file name by using the `--ssl-ca` server startup option.
- **ssl_cert**: The file that contains the server's digital certificate. The default value is `server-cert.pem`. Change the file name by using the `--ssl-cert` server startup option.
- **ssl_key**: File for the server's private key. The default value is `server-key.pem`. Change the file name by using the `--ssl-key` server startup option.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Example, displaying the current name of the file in the data directory that contains the list of trusted Certificate Authorities:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'ssl_ca';
+-----+-----+
| Variable_name | Value   |
+-----+-----+
| ssl_ca        | ca.pem |
+-----+-----+
1 row in set (0.00 sec)
```

SSL Client Options for Digital Certificates

If you REQUIRE a user account to use X509, ISSUER, or SUBJECT, the client must use both of the following options when initiating the connection:

- **--ssl-cert**: File name of the digital certificate issued to the client. Provides the identity of the client and public key.
- **--ssl-key**: File name of the private key for the client to use with its public key

Example:

```
# mysql -u root -p --ssl-cert=client-cert.pem --ssl-key=client-key.pem
```

Optionally, the client can provide details of the CA:

- **--ssl-ca**: File name containing the name of the CA that issued the server digital certificate



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Securing a Remote Connection to MySQL

- MySQL supports secure shell (SSH) connection to a remote MySQL server. This requires:
 - An SSH client on the client machine
 - Port forwarding through an SSH tunnel from the client to the server
 - Example: Forward requests from port 33306 on the local host to 3306 on the remote host
- A MySQL client application on the machine with the SSH client
 - Example: Run the mysql client on the local machine through the SSH tunnel.

```
# ssh -4 -L 33306:remotehostIP:3306 sshuser@remotehost
```

```
# mysql -u user -p -P33306 -h127.0.0.1
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Quiz



Which of the following options starts the MySQL server with SSL/TLS disabled?

- a. --ssl-mode=DISABLED
- b. --ssl-cipher=DHE-RSA-AES256-SHA:AES128-SHA
- c. --ssl=0
- d. RESET



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: c

Topics

- Security Risks
- Network Security
- Secure Connections
- **Password Security**
- Operating System Security
- Protecting Against SQL Injections
- MySQL Enterprise Firewall



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Preventing MySQL Password Security Risks

- Attackers use a number of techniques, including social engineering and key logging, to discover passwords.
 - Consider expiration policies to limit exposure if passwords are compromised.
- Attackers use social engineering to try to guess passwords.
 - Consider using the `validate_password` component to enforce a password policy that makes passwords more difficult to guess.
- Attackers try to find passwords in system tables and files.
 - MySQL passwords are encrypted by using a one-way hash and stored within the `mysql.user` table.
 - Prevent nonadministrative users from reading this table.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

How Attackers Derive Passwords

Attackers can derive plain text passwords from hashed passwords by using the following techniques:

- **Brute force algorithms** perform the hashing algorithm on many combinations of characters to find matching hashes.
 - These attacks are very slow and require large amounts of computation.
- **Dictionary attacks** perform hashing operations on combinations of dictionary words and other characters.
 - These are fast if the password is not secure.
- **Rainbow tables** are made up of the first and last hashes in long chains of repeatedly hashed and reduced passwords.
 - When you run a target password hash through the same algorithm chain and find a match to the end of a stored chain, you can derive the password by replaying that chain.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Even though passwords are stored as hashed values, attackers can still try to figure out passwords that match the hashed values. Do not grant non-administrative users access to the password tables or the operating system files that contain the hashed password values.

Password Validation Plugins and Components

MySQL uses plugins and components for authentication.

- `caching_sha2_password` is the default authentication plugin.
 - implements SHA-256 authentication, but uses server-side caching for better performance and provides additional features for wider applicability
- `mysql_secure_installation` installs the `validate_password` component.
 - enables extra system variables, including:
 - `validate_password.number_count`: number of numeric characters required
 - `validate_password.mixed_case_count`: number of uppercase and lowercase letters required
 - `validate_password.special_char_count`: number of special characters required
 - `validate_password.length`: minimum number of characters required in the password
 - Length has to be at least number count + special char count + 2 times the mixed-case count.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Other Variables for the Password Validation Component

- The `validate_password.policy` variable determines which of the `validate_password.xxx` variables are checked when a password is set or changed.
 - **0 or LOW:** Length
 - **1 or MEDIUM:** Length, numeric, upper/lower case, special characters
 - **2 or STRONG:** Length, numeric, upper/lower case, special characters, dictionary file
- If the policy is set to **STRONG**, the `validate_password.dictionary_file` variable must be set to point to a file of words to be checked.
- The `validate_password.check_user_name` variable is **ON** by default; it rejects a password that is the same as the user name or its reverse.
 - This option is not affected by the setting of the policy variable.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The password validation tests happen only when a password is being set or changed in a `CREATE USER` or `ALTER USER` statement. User accounts with passwords set before the policy is put in place can still log in and connect with passwords that do not meet the password policy criteria.

The dictionary file contains a list of words, one word per line, that is checked when the password policy is set to 2 or **STRONG**. Each substring of the password of length 4–100 is compared to the words in the dictionary file. The comparisons are not case-sensitive.

The `validate_password_check.user_name` variable is independent of the policy setting. If the password is set to the username of the current session or its reverse, the password is rejected.

Installing the Password Validation Component

- Installed by default when you use the Yum or SLES repositories or an RPM file to install MySQL
- Manual installation steps:
 1. Ensure that the component library file (`component_validate_password.so`) is located in the directory referenced by the `plugin_dir` server variable.
 2. Execute the following SQL statement:

```
INSTALL COMPONENT 'file:///component_validate_password';
```

3. Loads the component
4. Registers it in the `mysql.component` system table so that it loads automatically when the server is restarted

- Uninstall the component with the `UNINSTALL COMPONENT` statement:

```
UNINSTALL COMPONENT 'file:///component_validate_password';
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In Windows systems, the Validate Password component library file is `component_validate_password.dll`.

Validate Password Component Variables

```
mysql> SHOW VARIABLES LIKE 'validate%';
+-----+-----+
| Variable_name          | Value   |
+-----+-----+
| validate_password.check_user_name | ON      |
| validate_password.dictionary_file |        |
| validate_password.length       | 8       |
| validate_password.mixed_case_count | 1       |
| validate_password.number_count | 1       |
| validate_password.policy      | MEDIUM  |
| validate_password.special_char_count | 1       |
+-----+-----+
7 rows in set (#.## sec)
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When you install the Validate Password component, you can access the variables shown in the slide. If the password validation plugin is not installed, the variables do not appear.

The slide shows the default value of these variables.

Changing the Default Password Validation Variables

- You can dynamically set any of the password validation variables:

```
mysql> SET GLOBAL validate_password.policy = 2;
mysql> SET GLOBAL validate_password.length = 16;
```

- To persist the variable settings across server restarts, add them to a config file:

```
[mysqld]
validate_password.policy = 2
validate_password.length = 16
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Other Password Considerations

- If you do not use the password validation component, then assign a strong password to the `root` user.
 - The `root` account has full privileges for any database operation, and only trusted users should be able to access it.
- You can force all passwords to expire after a specified period, by setting the value of the `default_password_lifetime` system variable.

```
SET GLOBAL default_password_lifetime = number_of_days
```

— The default value is zero, which means that passwords *never* expire.

- You can set the password expiry time for a specific user in a `CREATE USER` or `ALTER USER` statement with the `PASSWORD EXPIRE` clause:

```
PASSWORD EXPIRE [DEFAULT | NEVER | INTERVAL n DAY]
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If a user tries to log in after the password has expired, the server might disconnect the user or put the user in sandbox mode, which limits the statements that the user can submit. In sandbox mode, the user can change the password with an `ALTER USER` statement, but cannot perform any other database operations. Whether a user is disconnected by the server or put into sandbox mode depends on client and server settings. If the client is able to handle expired passwords, or if, on the server, the `disconnect_on_expired_password` server variable is disabled, the server puts the client in sandbox mode. If the client cannot handle expired passwords, and the `disconnect_on_expired_password` variable is enabled (the default), then the client is disconnected with a message that the password has expired.

Locking an Account

- Lock individual accounts with the **ACCOUNT LOCK** clause in a `CREATE USER` or `ALTER USER` statement.
 - You might lock a new account when you initially create it with `CREATE USER` and unlock it when the user is ready to use it.
 - You might lock an existing account with `ALTER USER` if you suspect that it is compromised.
- View the lock state in the `mysql.user` table's `account_locked` column:

```
mysql> SELECT user, host, account_locked FROM mysql.user;
+-----+-----+-----+
| user | host | account_locked |
+-----+-----+-----+
| root | localhost | N |
| mysql.sys | localhost | Y |
+-----+-----+-----+
2 rows in set (#.## secs)
```

- Unlock a locked account by using the **ACCOUNT UNLOCK** clause.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Note that the `mysql.sys` account is locked. It cannot be used to log in. The `mysql.sys` account is the DEFINER of the `sys` schema objects. Roles are created as locked accounts.

Pluggable Authentication

- When a client connects to MySQL, the server uses the user name provided by the client and the client's host name to identify the appropriate row in the `mysql.user` table.
- The `plugin` column of the `mysql.user` table specifies which plugin to authenticate the user with.
 - enables different authentication mechanisms for different accounts: *pluggable authentication*
 - specifies for an account by using the `IDENTIFIED WITH method` clause of `CREATE USER` or `ALTER USER`
- The default authentication plugin is `caching_sha2_password`
 - modifies by setting the `default_authentication_plugin` system variable
- If you require other authentication methods that store their credentials somewhere other than the `mysql.user` table, install the appropriate plugin.
 - examples include: PAM, Windows login IDs, LDAP, or Kerberos
 - requires installing the server-side and client-side version of the plugin (if not built in)



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can specify that a particular user account should use another authentication plugin.

For example, if a user account is using the default `caching_sha2_password` plugin, the value in the `authentication_string` column is the SHA-256 encrypted value of the password, and the value in the `plugin` column is `caching_sha2_password`. To force an account to use the older `mysql_native_password` plugin, use the `IDENTIFIED WITH mysql_native_password BY 'newpassword'` clause. As a result of that command, in the `mysql.user` table, the `plugin` column for the user account is `mysql_native_password`, and the `authentication_string` column is the hashed value of the password.

To use PAM authentication, include `IDENTIFIED WITH authentication_pam AS 'authentication_string'`. In this case, the value stored in the `authentication_string` column is interpreted by the Pluggable Authentication Module as a service name or LDAP name to use for authentication rather than a hashed password.

The Windows authentication plugin is similar. To use Windows authentication, include `IDENTIFIED WITH authentication_windows AS 'authentication_string'`. In this case, the `authentication_string` is a Windows user or group and an optional map to a MySQL user account. For PAM authentication and Windows native authentication, MySQL depends on the external entity to authenticate the users and maintain passwords.

Preventing Application Password Security Risks

If you store application-specific user information in MySQL:

- Do not store plain text passwords in the database.
 - Store these passwords by using one-way hashes.
 - If you use plain text passwords and the application becomes compromised, an intruder can take the full list of passwords and use them.
- Use MySQL's `MD5()`, `SHA1()`, or `SHA2()` functions and store the password's hash value.
 - Alternatively, use some other one-way hashing function available to the application.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Connection-Control Plugin

- enforces a delay after a specified number of consecutive failed connection attempts
 - The delay increases with each consecutive failed connection after that number of attempts
- acts as a deterrent to brute force attacks
 - The more the failed connection attempts, the slower the server responds to subsequent attempts.
- exposes the following system variables:
 - `connection_control_failed_connections_threshold`: The number of successive failures permitted before a delay is added
 - `connection_control_min_connection_delay`: Amount of delay in milliseconds to add for each consecutive connection failure. The delay is this value multiplied by the number of failed connection attempts above the threshold.
 - `connection_control_max_connection_delay`: Maximum delay to add



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Set the appropriate system variables to specify the threshold for successive failed attempts allowed, the amount of delay, and the maximum delay. If the threshold is set to 3 and the amount of delay is set to 1000, the fourth successive failed attempt (one above the threshold) causes a delay of 1000 milliseconds. The fifth failed attempt causes a delay of 2000 milliseconds and so on until the maximum delay value is reached.

Installing the Connection-Control Plugin

- Install the plugin:

```
INSTALL PLUGIN connection_control SONAME 'connection_control.so';
```

- View its configuration variables:

```
mysql> SHOW VARIABLES LIKE 'connection_control%';
+-----+-----+
| Variable_name          | Value   |
+-----+-----+
| connection_control_failed_connections_threshold | 3      |
| connection_control_max_connection_delay           | 2147483647 |
| connection_control_min_connection_delay           | 1000    |
+-----+-----+
3 rows in set (#.## sec)
```

- Set the variable values dynamically or within a config file.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

For Windows systems, in the statement to install the plugin, replace 'connection_control.so' with 'connection_control.dll'.

If the plugin is not installed, the variables do not appear. The default threshold is 3, the default delay value is 1000, and the default max delay is 2147483647.

Monitoring Connection Failures

- Inspects the value of the **Connection_control_delay_generated** status variable.
 - counts the number of times the server added a delay for a failed connection attempt
 - example:

```
mysql> SHOW STATUS LIKE 'Connection_control%';
+-----+-----+
| Variable_name          | Value |
+-----+-----+
| Connection_control_delay_generated | 7    |
+-----+-----+
1 row in set (#.## sec)
```

- considers installing the **CONNECTION_CONTROL_FAILED_LOGIN_ATTEMPTS** plugin
 - creates a table in the Information Schema to maintain more detailed information about failed connection attempts
 - The Connection-Control plugin populates the table.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Using the CONNECTION_CONTROL_FAILED_LOGIN_ATTEMPTS Plugin

- Install the plugin using the same file name as the Connection-Control plugin:

```
mysql> INSTALL PLUGIN CONNECTION_CONTROL_FAILED_LOGIN_ATTEMPTS SONAME  
'connection_control.so';
```

- Query the Information Schema's CONNECTION_CONTROL_FAILED_LOGIN_ATTEMPTS table.
 - Its columns identify the user account and the number of failed connection attempts.

```
mysql> SELECT * FROM  
      -> information_schema.CONNECTION_CONTROL_FAILED_LOGIN_ATTEMPTS;  
+-----+-----+  
| USERHOST          | FAILED_ATTEMPTS |  
+-----+-----+  
| 'employee'@'localhost' |      8 |  
| 'root'@'localhost'   |      7 |  
+-----+-----+  
2 rows in set (#.## sec)
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

For Windows systems, in the statement to install the plugin, replace 'connection_control.so' with 'connection_control.dll'.

In this example, one user had 8 failed attempts and the other user had 7 failed attempts. If the threshold is set to 4, then one user would have had 4 attempts delayed and the other user would have had 3 attempts delayed, making the value of Connection_control_delay_generated status variable $7 - (4 + 3)$.

Quiz



Which of the following validations is included **only** when the validate_password_policy variable is set to 2 or STRONG?

- a. Numbers are required.
- b. Words are compared to a dictionary file.
- c. Special characters are required.
- d. Passwords must use mixed case letters.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: b

Topics

- Security Risks
- Network Security
- Secure Connections
- Password Security
- **Operating System Security**
- Protecting Against SQL Injections
- MySQL Enterprise Firewall



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

LIMITING OPERATING SYSTEM USAGE

- Minimize the number of OS accounts on the MySQL host.
 - You normally administer MySQL by using a login account dedicated to that purpose.
 - Other accounts increase the number of possible attack vectors on the host.
 - Login accounts are not necessary for MySQL-only machines.
- Minimize the number of non-MySQL-related tasks on the server host.
 - Additional services might open additional ports and create additional attack vectors.
 - When you configure a host for fewer tasks, it can be more easily secured than a host running a complex configuration that supports many services.
 - Dedicating a system to MySQL provides performance benefits.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Limiting Operating System Accounts

- Restrict the number of users who can access the host.
 - Each additional login increases the risk of exposing database information that belongs to the MySQL installation and its administrative account.
 - Examples
 - Improper file system privileges can expose data files.
 - Users can run the `ps` command to view information about processes and their execution environment.
- When you use a machine dedicated to MySQL, use only the following accounts:
 - system administrative accounts (such as `root` in Linux or user-specific accounts that can use `su` or `sudo`)
 - accounts that might be needed for administering MySQL itself (such as the `mysql` user account)



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Operating System Security

- For multi-user systems such as Linux, set ownership of all components of a MySQL installation to a dedicated login account with minimal privileges.
 - Typical installations use the `mysql` account.
- This protects the database directories from access by users who are not responsible for database administration.
- An additional benefit of setting up this account is that it can be used to run the MySQL server, rather than running the server from the Linux `root` account.
- A server that has the privileges of the `root` login account has more file system access than necessary and constitutes a security risk.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

MySQL Service Security

- Do not grant access to the user table in the `mysql` database.
 - The `root` user has access.
- Put MySQL behind the firewall or in a demilitarized zone (DMZ).
- Before starting the server and setting passwords, take any actions necessary to protect MySQL-related portions of the file system.
 - If you set passwords before you protect the files, someone with direct file system access on the server host might replace the files.
 - This compromises the MySQL installation and undoes the effect of setting the passwords.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

File System Security

- Protect MySQL files from being accessed by other users on the file system.
 - Data directories
 - InnoDB tablespaces
 - Backup files
 - Configuration files that contain plain text or encrypted passwords
 - my.cnf or mylogin.cnf files
- A user who gains access to MySQL data files or backups can restore those files to databases on another server.
- A MySQL installation also includes the programs and scripts used to manage and access databases.
 - Users need to be able to run but not modify some of these (such as the client programs).



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Preventing File System Security Risks

- Change data directory ownership and access permissions before starting the server.
 - Assign file ownership to an account with administrative privileges.
 - Set MySQL-related directories and files and user and group table ownership to mysql, including:
 - MySQL programs
 - Database directories and files
 - Log, status, and configuration files
- Do not set passwords before protecting files. This can permit an unauthorized user to replace the files.
- Set up a dedicated system account for MySQL administration.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Topics

- Security Risks
- Network Security
- Secure Connections
- Password Security
- Operating System Security
- **Protecting Against SQL Injections**
- MySQL Enterprise Firewall



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Protecting Your Data from SQL Injection Attacks

Take measures to protect your data from application-based attacks, such as SQL injection.

- Do not trust any data entered by users of your applications.
 - Users can exploit application code by using characters with special meanings, such as quotes or escape sequences.
 - Make sure that your application remains secure if a user enters something like `DROP DATABASE mysql;`.
- Protect numeric and string data values.
 - Otherwise, users can gain access to secure data and submit queries that can destroy data or cause excessive server load.
- Protect even your publicly available data.
 - Attacks can waste server resources.
 - Safeguard web forms, URL names, special characters, and so on.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

SQL Injection: Example

- A badly written application compares a provided username and password with rows in the user table and ensures there is a single matching row with a line such as:

```
sql = "SELECT COUNT(*) FROM users WHERE user='"
+ username + "' AND pass = '" + password + "'";
```

- If the user enters a username and password of Peter and tY*wa8?L respectively, the statement evaluates as:

```
SELECT COUNT(*) FROM users WHERE user='Peter'
AND pass = 'tY*wa8?L'
```

- If the user enters a username and password of abcd and x' OR 1=1 LIMIT 1;-- respectively, the statement evaluates as:

```
SELECT COUNT(*) FROM users WHERE user='abcd'
AND pass = 'x' OR 1=1 LIMIT 1;-- '
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows an attack that permits logins for users who do not provide a matching password. SQL injection attacks might also include commands that create users, drop databases, or modify critical data.

Detecting Potential SQL Injection Attack Vectors

Users may attempt SQL injection by any of the following methods:

- entering single and double quotation marks (' and ") in web forms
- modifying dynamic URLs by adding %22 ("), %23 (#), and %27 (') to them
- entering characters, spaces, and special symbols rather than numbers in numeric fields

Ensure that the application generates an error or removes these extra characters before passing them to MySQL.

- If the application permits these characters, your application security might be compromised. Communicate this to the application developers.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Preventing SQL Injection Attacks

- Never concatenate user-provided text with SQL statements in the application.
- Use parameterized stored procedures or prepared statements when you perform queries that require user-provided text.
 - Stored procedures and prepared statements do not perform macro expansion with parameters.
 - Numeric parameters do not permit text values such as injected SQL syntax.
 - Text parameters treat the user-provided value as a string for comparison rather than SQL syntax.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Note: SQL injection attacks and how to prevent them are covered in the course titled *MySQL for Developers*.

Topics

- Security Risks
- Network Security
- Secure Connections
- Password Security
- Operating System Security
- Protecting Against SQL Injections
- MySQL Enterprise Firewall



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

MySQL Enterprise Firewall

- Plugin available with Enterprise Edition only
- Application-level firewall
- Permits or denies SQL statements
 - Registered accounts have whitelists of acceptable statement patterns.
- Operates in per-account modes:
 - **RECORDING**: identifying acceptable statements and recording their patterns in a whitelist
 - **PROTECTING**: preventing statements that do not match patterns in the whitelist
 - **DETECTING**: logging suspicious statements, but not preventing statements
 - **OFF**: does not record or protect statements. This is the default mode for each account.
- Can be disabled for trusted accounts



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Enterprise Firewall Plugins

- Plugin functions:
 - **MYSQL_FIREWALL**: examines statements and executes or rejects statements based on rules in its cache
 - **MYSQL_FIREWALL_USERS** and **MYSQL_FIREWALL_WHITELIST**: implements Information Schema tables that contain information about the firewall cache
- All three plugins are in the shared library file **firewall.so**.
 - distributed with Enterprise Edition
 - located in the `lib/plugin` directory in binary distributions



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In Windows installations, the shared library file is called `firewall.dll`.

Enterprise Firewall Database Components

- The `sp_set_firewall_mode()` stored procedure registers MySQL accounts with the firewall.
 - This is the only component that is intended for direct use.
- Other components include:
 - tables in the `mysql` database that store persistent copies of firewall cache data:
 - `firewall_users`: Registered users
 - `firewall_whitelist`: Whitelisted statement patterns for each user
 - library functions that are used internally by the firewall:
 - `set_firewall_mode()`
 - `normalize_statement()`
 - `read_firewall_whitelist()`
 - `read_firewall_users()`



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Installing MySQL Enterprise Firewall

- Use the correct installation script based on your operating system:
 - Linux and other systems that use .so shared libraries:
`linux_install_firewall.sql` installs the `firewall.so` plugin library.
 - Windows: `win_install_firewall.sql` installs the `firewall.dll` plugin library.
 - If you install MySQL Server by using the MySQL Installer, you can also elect to install MySQL Enterprise Firewall by checking the “Enable Enterprise Firewall” check box.
 - The installation scripts are located in the `share` subdirectory of the MySQL Server installation.
- The installer script:
 - installs the firewall plugins from the library file
 - creates the firewall configuration stored procedure, tables, and internally used function references in the `mysql` database



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Registering Accounts with the Firewall

Register an account by setting its initial firewall mode.

- The account name is in the full `user@host` format, stored as a single string.
- To register an account that is not initially controlled by the firewall, set the mode to `OFF`.

```
CALL mysql.sp_set_firewall_mode('appuser@apphost', 'OFF')
```

- To register an account for firewall training, set the initial mode to `RECORDING`.

```
CALL mysql.sp_set_firewall_mode('appuser@apphost', 'RECORDING')
```

- If you set an initial mode of `PROTECTING`, the account cannot execute any statements because its whitelist is empty.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Training the Firewall

- Register the account in **RECORDING** mode.
- The firewall creates a normalized statement digest for each statement and places the digest in the account's whitelist cache.
- Switch the mode to **PROTECTING** or **OFF** when training is complete to persist the whitelist.
 - The firewall persists the cache when you change the account's mode.
 - If you restart the `mysqld` process while in **RECORDING** mode, any changes to that account's whitelist cache are lost.
- Return to **RECORDING** mode to learn new statements if the application changes.
 - Changing mode from **OFF** or **PROTECTING** to **RECORDING** does not clear the account's whitelist.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Statement Digests

- The firewall transforms statements to statement digests before storing them in the whitelist.
 - condenses whitespace
 - Statements that differ only in whitespace are equivalent.
 - removes comments
 - replaces literal values with placeholders
- Example: If you execute the following two statements while in RECORDING mode, they record the same whitelisted digest:

```
INSERT INTO my_table VALUES (1, 'Alpha');  
INSERT INTO my_table VALUES (2, 'Beta');
```

- Whitelisted digest form of the preceding two statements:

```
INSERT INTO `my_table` VALUES (...)
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Enabling Firewall Protection

- Set the account to **PROTECTING** mode.
 - Accounts are protected only if they are registered with the firewall and are in **PROTECTING** mode.
- Statements that are not in digest form in the whitelist are prevented from executing.
 - The client receives an error message.

```
CALL mysql.sp_set_firewall_mode('appuser@apphost', 'PROTECTING')
```

- The server writes a message to the error log.

```
ERROR 1045 (28000) : Statement was blocked by Firewall
```

- The client receives an error message.

```
[Note] Plugin MYSQL_FIREWALL reported: 'ACCESS DENIED for user@host.  
Reason: No match in whitelist. Statement: Statement'
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Disabling the Firewall

- To disable the firewall for a single account, set its mode to **OFF**.

```
CALL mysql.sp_set_firewall_mode('appuser@apphost', 'OFF')
```
- To reset the whitelist for an account, set its mode to the special **RESET** mode.

```
CALL mysql.sp_set_firewall_mode('appuser@apphost', 'RESET')
```

- clears the whitelist
- sets the account's firewall mode to OFF
 - The **RESET** mode is used only to clear the whitelist and is not a settable mode.

- To disable the firewall for all accounts, set the `mysql_firewall_mode` dynamic global variable to **OFF**.
 - Set to **ON** by default.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Monitoring the Firewall

The Enterprise Firewall plugin adds status variables so that you can monitor:

- the number of statements that the firewall has denied
- the number of statements that the firewall has permitted
- the number of statements that were identified as being suspicious while in DETECTING mode, but still permitted
- the number of whitelisted digests in the cache

```
mysql> SHOW GLOBAL STATUS LIKE 'Firewall%';
+-----+-----+
| Variable_name          | Value |
+-----+-----+
| Firewall_access_denied | 3      |
| Firewall_access_granted | 4      |
| Firewall_access_suspicious | 1      |
| Firewall_cached_entries | 4      |
+-----+-----+
4 rows in set (0.00 sec)
```

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Quiz



Which of the following modes clears the Enterprise Firewall whitelist for a particular user?

- a. OFF
- b. PROTECTING
- c. DETECTING
- d. RESET



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: d

Summary



In this lesson, you should have learned how to:

- Recognize common security risks
- List security problems and counter-measures for networks, passwords, operating systems, file systems, and applications
- Protect your data from interception and access
- Use SSL for secure MySQL server connections
- Use SSH to create a secure remote connection to MySQL
- Configure and use MySQL Enterprise Firewall

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practices

- 7-1: Quiz—Securing MySQL
- 7-2: Enabling SSL for Secure Connections
- 7-3: Configuring MySQL Enterprise Firewall



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

8

Maintaining a Stable System

ORACLE®



MySQL™

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide

Objectives



After completing this lesson, you should be able to:

- Improve the stability of MySQL servers
- Monitor database growth and explain capacity planning
- Troubleshoot server slowdowns
- Identify locked resources
- Perform InnoDB crash recovery

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Topics

- Stability
 - Why Databases Fail
 - Capacity Planning
 - Troubleshooting
 - Identifying the Causes of Server Slowdowns
 - InnoDB Recovery



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Stable Systems

- Stable systems exhibit predictable behavior over a period of time.
 - Servers run without unplanned outages.
 - Planned outages are rare.
 - Applications exhibit predictable performance.
- Stability is difficult to maintain in a changing environment.
 - Applications change with the business.
 - Usage patterns change as the user base grows.
 - The environment changes as you upgrade hardware and operating systems.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Measuring What You Manage

- Establish a performance baseline to measure the system's normal variable values.
- After every configuration change, measure variables again and compare against your baseline.
 - Hardware and software upgrades
 - Exploratory configuration changes
 - Changes in the infrastructure
- Measure variables regularly to update the baseline.
 - Changes in application usage patterns
 - Data growth over time
- Whenever you encounter a problem, compare values with the baseline.
 - When you precisely define a problem, the solution often becomes obvious.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Establishing a Baseline

- Define what is normal:
 - The baseline is something to compare against when you encounter a problem.
 - Over time, changes in the baseline provide you with useful information for capacity planning.
- Record operating system metrics: file system, memory, and CPU usage
 - top, iostat, vmstat , sysstat , sar on Linux/Unix based systems
 - Resource Monitor and Performance Monitor on Windows
- Record MySQL status and configuration:
 - SHOW PROCESSLIST or sys.session to see running processes
 - mysqladmin extended-status to see status variables
 - Use -iseconds --relative to record value deltas.
- Profile application use-case response times:
 - Log in, search, create, read, update, and delete.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Keep a copy of the my.cnf file that the server used when you created the baseline so that you can identify when a configuration change results in a problem at a later time.

Application Profiling

- Records the time at which specific events occur
 - Function entry/exit
 - Calls to external systems, such as databases
- Measures how long each part of an operation takes
- Is integrated with many development environments and interpreters, or as plugins
 - Alternatively, you can instrument your code to profile significant points.
- Shows if it is useful to troubleshoot the performance of the database
 - Example: If a database call makes up 5% of the time it takes to perform a function, such as logging in or searching, you might gain more performance by troubleshooting another part of that function.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When you record an application profile as part of your baseline, you can see the duration of key parts of each function or use case. This enables you to see if the application is experiencing most of its delay on calls to the database, setting up connections, or if the delay is due to some other application operation, such as internal memory allocation, a sort function, or calls to some other external process.

Topics

- Stability
- Why Databases Fail
- Capacity Planning
- Troubleshooting
- Identifying the Causes of Server Slowdowns
- InnoDB Recovery



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Asking “What Could Go Wrong?”

- Consider all the components in an architecture:
 - Servers (database and application)
 - Storage
 - Network interfaces
 - Power supplies, memory, CPU
 - Connectivity
 - Networking infrastructure
 - Firewalls
 - Load balancing
 - Application software
 - User-facing components
 - Framework stability
- Consider *force majeure*:
 - Natural disasters or other extraordinary events

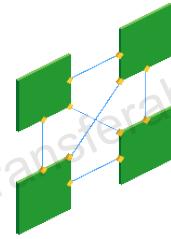


ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Components in a MySQL Server Installation

- Physical environment
 - Server power supply
 - Physical security
 - Server hardware
- Virtualized environment
- Operating system
- MySQL process and components
- Coexistent services
- Network infrastructure
- Connected applications



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Server Hardware

- The server room is an important part of a database environment.
 - Ensure it is secure and stable, whether it is a closet or a large data center facility.
- Mitigate server failure risks by having redundant hardware components.
 - Power supplies
 - RAID in any fault-tolerant configuration
 - Network adaptors
- In the most common server architectures, other components such as RAM and CPUs are potential points of failure.
 - Mitigate such risks by maintaining and testing server failover plans.



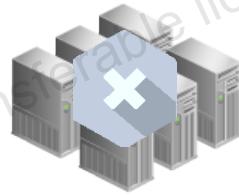
ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

RAID 0 is not fault tolerant. Other RAID configurations in common use (1, 5, 6, and nested levels such as 10) are fault tolerant.

Problems with Hardware

- Problems caused by concurrent activity on the hardware are often intermittent.
 - They might occur at similar times daily or weekly.
- These problems might be difficult to diagnose.
 - Their causes are often independent of the application that manifests the problem.
- Degraded hardware performance can result in overall system issues.
 - They might be difficult to detect without investigating the system logs.
 - Example: A degraded RAID set continues to perform I/O and serve requests with no application-visible symptoms other than lower I/O performance.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Virtualized Environment

If MySQL runs in a virtual machine (VM) within a virtualization platform, that platform becomes a component with attached risk.

- Shares some of the resources of the host system between guests
 - Hard disk
 - Memory
 - CPU allocation
 - Network interfaces
- Might be vulnerable to resource contention
 - Dedicate hardware resources where possible.
 - Hard disks, CPUs, network interfaces
 - Extend application timeout tolerances to account for allocation delays caused by other guest systems or the virtualization platform.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Operating System

- The operating system on which MySQL runs is a vital component in its architecture.
 - An operating system failure becomes a database failure.
- Ensure that the operating system is maintained and stable.
 - Apply security patches and updates.
- Consider the operating system's performance and security mechanisms and how they impact MySQL.
 - File systems and storage
 - Mandatory access control, for example, SELinux
- Monitor the operating system's logs and variables.
 - This is particularly important if the server is not dedicated to MySQL but is shared with other services or applications.
 - Example: Ensure the file system does not fill up with MySQL data or logs, or logs from other services.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Coexistent Applications

Applications that share a server with MySQL come with associated risks.

- Security:
 - Breaches in another application on the same server might permit attackers to access other files including data files on the server.
 - Bugs in an application might result in performance degradation of the server or inappropriate file system access.
- Performance:
 - Applications on the same server as MySQL share resources and might impact MySQL's performance.
 - CPU, memory, I/O, network bandwidth



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Network Failures

- MySQL communicates over the network in a number of ways:
 - Client connections from applications
 - Replication with other MySQL servers
 - Administrative connections
 - Monitoring software
 - MySQL Enterprise Monitor
- Other network activity might interfere with MySQL:
 - Operating system backups over the network
 - Application traffic
 - File transfer and other network services
- Ensure that network hardware does not become a single point of failure.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Application Failures

- Many performance problems result from the application code rather than the database server. Examples include:
 - Reading large data files
 - Calling remote web services
 - Inefficient sorting or searching algorithms against large data sets
- Use application profiling to identify possible performance problems in the application before assuming they are related to the database.
 - Measure the duration of each application function by using an IDE or external profiler.
- Bugs in the application might create incorrect data or permit security breaches.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Force Majeure

Extraordinary events might result in the loss of assets in (or access to) a location.

- Plan for disaster by considering disaster recovery and data center failover.
- Mitigate the risk of loss by:
 - Maintaining a well-tested backup strategy
 - Practicing failover to servers in an alternative location
 - Maintaining an active data center in a separate physical location



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Topics

- Stability
- Why Databases Fail
- **Capacity Planning**
- Troubleshooting
- Identifying the Causes of Server Slowdowns
- InnoDB Recovery



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Capacity Planning

- The system must have the capacity to handle any projected usage growth and any transient spikes in activity.
 - Consider changes in baseline resource usage due to user activity and data growth.
 - Consider upcoming campaigns or other predicted busy times.
- Do not add too many resources at one time.
 - To add servers (or other hardware) that you do not need is wasteful and does not provide a worthwhile return on investment.
- Graph key elements in your baseline to monitor changes in resources that grow in usage as your data or application functionality grows:
 - Memory
 - Hard disk space

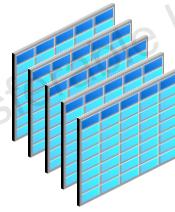


ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Monitoring Table Size

- InnoDB row and index data are stored in data pages.
 - The default page size is 16 KB.
 - InnoDB tables and indexes consist of leaf pages containing data, and nonleaf pages containing page pointers.
 - Pages often have free space because:
 - InnoDB orders rows according to primary key
 - InnoDB keeps columns for each row on the same page
 - Exceptions: Variable length columns, such as VARCHAR and BLOB, that exceed the row size might be stored in overflow pages.
 - Rows do not always evenly fill data pages.
- Logical table size is smaller than physical file size.
 - Logical size includes data and index pages.
 - In addition, physical size includes nonrow file content.
 - Empty pages, header and footer information



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Calculating Logical Size: Data and Indexes

Select the DATA_LENGTH and INDEX_LENGTH columns from INFORMATION_SCHEMA.TABLES for each table.

```
mysql> SELECT TABLE_NAME AS `table`,  
-> DATA_LENGTH + INDEX_LENGTH AS `logical size`  
-> FROM INFORMATION_SCHEMA.TABLES  
-> WHERE TABLE_SCHEMA='employees';  
+-----+-----+  
| table | logical size |  
+-----+-----+  
| departments | 32768 |  
| dept_emp | 22593536 |  
| dept_manager | 49152 |  
| employees | 15220736 |  
| salaries | 136511488 |  
| titles | 31571968 |  
+-----+-----+  
6 rows in set (#.## sec)
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can also execute SHOW TABLE STATUS LIKE 'tablename' to view the Data_length and Index_length columns for each table, but you cannot sum them in the same query.

Calculating Physical Size: Querying Information Schema

- The INFORMATION_SCHEMA.FILES view contains information about InnoDB tablespaces:

```
mysql> SELECT FILE_NAME, TOTAL_EXTENTS * EXTENT_SIZE as `size`  
->   FROM INFORMATION_SCHEMA.FILES  
-> WHERE FILE_NAME LIKE '%employees%';  
+-----+-----+  
| FILE_NAME          | size    |  
+-----+-----+  
| ./employees/employees.ibd | 23068672 |  
| ./employees/departments.ibd |      0 |  
| ./employees/dept_manager.ibd |      0 |  
| ./employees/dept_emp.ibd | 30408704 |  
| ./employees/titles.ibd | 41943040 |  
| ./employees/salaries.ibd | 146800640 |  
+-----+-----+  
6 rows in set (0.00 sec)
```

- Tablespaces that contain only partially filled extents are not included.
 - The departments and dept_manager tables each contain less than EXTENT_SIZE (1 MB by default).



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

This technique applies to all tablespaces. For tablespaces that contain multiple tables, such as the system tablespace and general tablespaces, the size shown in the output is the sum of extents used by all tables.

Calculating Physical Size: Reading the File System

View the size of the .ibd file containing that table's data.

```
# cd /var/lib/mysql/employees
# ls -l *.ibd
-rw-r----- 1 mysql    114688 Jun 24 20:00 departments.ibd
-rw-r----- 1 mysql   30408704 Jun 24 20:00 dept_emp.ibd
-rw-r----- 1 mysql   131072 Jun 24 20:00 dept_manager.ibd
-rw-r----- 1 mysql  23068672 Jun 24 20:00 employees.ibd
-rw-r----- 1 mysql 146800640 Jul 27 11:07 salaries.ibd
-rw-r----- 1 mysql  41943040 Jun 24 20:00 titles.ibd
```

This technique requires file-per-table tablespaces.

- It depends on the `innodb_file_per_table` option, which is enabled by default.
- If enabled, general tablespaces and the system tablespace store multiple tables in each file.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Scalability

A scalable system:

- Enables proportional performance increases when you add hardware or other resources
 - Additional RAM or network bandwidth
 - Additional servers
- Provides predictable throughput and query results when you increase the load
 - Number of concurrent users
 - Quantity of data requested
- Facilitates capacity planning
 - Current and near-future demands
 - Cost-effective: Capacity to enable growth without being prohibitively expensive



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Scaling Up and Scaling Out

- Scaling up:
 - Add more CPU, storage, or main memory resources to increase the processing capacity of any single node.
 - In general, scaling up is less complicated than scaling out because of the difficulty in writing software that performs well in parallel.
- Scaling out:
 - Add more servers to the environment to enable more parallel processing.
 - Software (application or storage engine) needs to be written to use multiple locations.
 - Examples:
 - Sharded database, replication for analytics or backups, InnoDB Cluster, NDB storage engine in MySQL Cluster



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

A sharded database is a system that uses multiple database servers, each one containing a defined subset of the overall data. Clients must access the correct server when they store or retrieve data, based on the sharding key.

Quiz



Which of the following activities would qualify as scaling out?

- a. Adding a redundant power supply on a second uninterruptable power supply to a server to prevent power loss
- b. Adding a second server containing a read-only copy of data to drive some webpages
- c. Configuring RAID 10 storage on multiple disks on a server to improve performance and redundancy
- d. Installing a second MySQL instance on the same server, on port 3307



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: b

Topics

- Stability
- Why Databases Fail
- Capacity Planning
- **Troubleshooting**
- Identifying the Causes of Server Slowdowns
- InnoDB Recovery



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Establishing the Nature of a Problem

To identify a problem, answer the following questions:

- Has the application, database, or server configuration changed recently?
 - Or were changes made but not persisted on server restart?
- Has the problem resolved itself since it first occurred?
 - Was there a sudden growth of application activity due to a batch operation or a spike in web traffic?
 - Were system resources taken up by operations that are external to the database?
 - Network traffic causing router problems
 - File system backups causing I/O problems
- Does the problem occur at predictable intervals?
 - At a regular time of the day or week
 - During or after some repeatable operation



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

A frequent first sign of a performance problem is a user-visible application error. To find the cause of the error, you must track through the components from the application to the database to determine where the problem lies.

Problems can also be caused by factors that are distinct from the database and application. If a network router or switch crashes or becomes overloaded due to a large amount of traffic, communications between the application and database can be interrupted. Similarly, a large amount of hard disk traffic (such as that caused by a file system backup) can interrupt I/O operations.

Identifying the Problem

- Compare application, MySQL, and OS settings and other metrics with the baseline.
 - Re-execute the tests used to create the baseline.
- Localize the problem at a functional level.
 - Identify what is manifesting the problem.
 - Specific application use cases
 - Specific clients
- Create a clear problem statement.
 - Error messages
 - Specific behavioral changes
 - Intermittent or continuous symptoms
 - Reproducible symptoms



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You must localize the problem at a functional level. For example, if logging in is much slower than before, but the rest of the application is working correctly, that problem is likely to have a very different root cause than the search function being slow. Similarly, if every application function slows down every afternoon at 2:30 PM, that problem is likely to have yet another cause.

Common Problems

- The most common problems occur when you change the configuration.
 - If you change a configuration file and use invalid settings
- A change in usage patterns can affect the performance and stability of the database.
 - Example: You experience large increases in data volume or traffic.
- Performance problems such as application timeouts might appear to resolve themselves after a period of time.
 - Such problems might be caused by:
 - Sudden increases in activity from batch operations
 - Highly publicized campaigns
 - Webpages that go viral and encounter a much larger amount of traffic than usual



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Resolving Problems

- Problems with misconfiguration are usually easy to resolve after they are identified:
 - The server fails to start and the error log contains the reason.
 - Performance degrades after a restart.
 - Ensure that you record configuration changes so that you can easily undo them.
- Resolve performance problems by doing the following:
 - Improve the database structure (schema and indexes).
 - Improve the local database server environment (scale up).
 - Network, operating system, server performance, and memory
 - Improve the networked database architecture (scale out).
 - Sharding, replication, MySQL Cluster
 - Fine-tune database settings.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

MySQL's settings have default values that apply to most environments. There are no "easy fixes" that improve a typical database's performance, because the engineers who create MySQL already design performance into the default settings. In any specific environment, it might be possible to gain a slight improvement in performance by tuning MySQL settings if you have already optimized the database schema, indexes, server platform, and application architecture.

Note: Fine-tuning database settings is covered in the course titled *MySQL Performance Tuning*.

Topics

- Stability
- Why Databases Fail
- Capacity Planning
- Troubleshooting
- **Identifying the Causes of Server Slowdowns**
- InnoDB Recovery



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Identifying the Causes of Server Slowdowns

Start your investigations by focusing on the most common problems and their typical causes.

- A small number of queries:
 - Query plans
 - Locks caused by other queries on the same table
- Many (or all) queries:
 - General server activity, contention, and mutexes
- Many queries on a single table:
 - Schema design, table indexes, and statement structure
- Intermittent or consistent performance problems:
 - Consider the concurrent load when problems occur.
 - Examine transactions that occur at that time.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Investigating Slowdowns

- When a problem affects a few queries:
 - Look at the indexes in the table, the design of those statements, or the design of your schema.
- When a problem affects many (or all) queries:
 - Examine configuration of the database server, its internal activity, or its interaction with its environment.
 - Look for contention on a low-level MySQL resource.
 - Look for a problem with the volume of I/O generated by MySQL.
- There might be multiple causes for a general server performance degradation.
 - Example: If multiple queries on different tables are slower than usual, the problem could come from two or more problems with poorly performing queries.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Quiz

Q

Which of the following activities improves database efficiency the most in a typical (and otherwise unoptimized) MySQL configuration?

- a. Add RAID 10 storage to the server.
- b. Configure replication.
- c. Optimize the database schema and indexes.
- d. Optimize the server settings.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Answer: c

How MySQL Locks Rows

MySQL locks resources in the following ways:

- Server-level data locks:
 - Table locks
 - Metadata locks
- Storage engine data locks:
 - Row-level locks
 - Handled at the InnoDB layer
- Mutexes:
 - Lower-level locks that apply to internal resources rather than to data
 - Examples: Log files, AUTO_INCREMENT counters, and InnoDB buffer pool mutexes
 - Used for synchronizing low-level code operations, ensuring that only one thread can access each resource at a time



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Identifying Lock Contention

- Understand how InnoDB locks rows.
- Identify long running or blocked queries with `SHOW PROCESSLIST`.
- Identify mutex contentions by querying for synch (/wait/synch/mutex/*) instruments in the Performance Schema `events_waits_%` tables.
- Identify blocking and waiting transactions by querying Information Schema and Performance Schema views.
- Identify current locks by querying the Performance Schema `data_locks` table.
- Identify metadata locks by querying the Performance Schema `metadata_locks` table.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

InnoDB Table Locks

- Types of InnoDB table-level locks
 - **Shared (S)**: Locks the table for read.
 - **Exclusive (X)**: Locks the table for write.
 - **Intention Shared (IS)**: Locks the table to allow shared row-level locking.
 - **Intention Exclusive (IX)**: Locks the table to allow exclusive row-level locking.
- A transaction can acquire a lock when another transaction holds a lock only if their lock types are compatible. Table lock type compatibility matrix:

	X	IX	S	IS
X	Conflict	Conflict	Conflict	Conflict
IX	Conflict	Compatible	Conflict	Compatible
S	Conflict	Conflict	Compatible	Compatible
IS	Conflict	Compatible	Compatible	Compatible



InnoDB Row Locks

- A transaction must acquire a table intention lock (**IS** or **IX**) before locking rows.
 - Shared (**S**) :
 - Permits the transaction to read a row (also called **READ** mode)
 - Allows other transactions to acquire shared locks on the same row but not exclusive lock
 - Exclusive (**X**):
 - Permits the transaction to read and write a row (also called **WRITE** mode)
 - Does not allow any other transaction to lock the row
- Examples:
 - If transaction *A* holds an exclusive lock on a row, and transaction *B* requests a shared lock on that row, those locks are in conflict and transaction *B* becomes blocked waiting for that lock.
 - If transaction *C* holds a shared lock on a row and transaction *D* requests a shared lock on the same row, those locks are compatible and transaction *D* acquires the lock.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Troubleshooting Locks with SHOW PROCESSLIST

Look for waits in the State column

- The table has a conflicting lock:

State: Waiting for table metadata lock

- An InnoDB row (or table) lock:

State: update

or:

State: Searching rows for update

- The server (through SHOW PROCESSLIST) does not display internal InnoDB information about locks.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Process information is also available from the performance_schema.threads table (PROCESSLIST_STATE field). However, access to threads does not require a mutex that is required by SHOW PROCESSLIST and has minimal impact on server performance.

SHOW PROCESSLIST: Example

```
mysql> SHOW PROCESSLIST\G
***** 1. row *****
    Id: 14
    User: root
    Host: localhost
    db: world
Command: Query
    Time: 15
  State: Waiting for table metadata lock
    Info: INSERT INTO City VALUES (5000, 'Galway', 'IRL', 'Connaught', '80000')
...
...
```

- The value in the **State** column shows that another process has locked the table.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Monitoring Data Locks with Information Schema and Performance Schema

- Information Schema and Performance Schema contain views that link blocked transactions to the transactions that hold those locks.
 - **INFORMATION_SCHEMA.INNODB_TRX**: General transaction information and lock status
 - **performance_schema.data_locks**: Information about each lock and the locked resource
 - **performance_schema.data_lock_waits**: The blocked transaction and the transaction that is blocking it
- Each view contains information about some part of the relationship between blocking and blocked statements.
 - By joining the views, you can see which transaction is blocking another.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Information Schema INNODB_TRX View

- General information about running transactions
 - Transaction ID: `trx_id`
 - Thread ID: `trx_mysql_thread_id`
 - Transaction start time: `trx_started`
 - Transaction query that is currently being executed: `trx_query`
 - `NULL` if the transaction is idle
 - Transaction state: `trx_state`
- Lock status
 - The lock ID of the lock that is currently blocking the transaction, if any: `trx_requested_lock_id`
 - The time that the current lock was requested: `trx_wait_started`

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_TRX\G
***** 1. row *****
      trx_id: 1510
      trx_state: RUNNING
      trx_started: date-and-time
      trx_requested_lock_id: NULL
      trx_wait_started: NULL
      trx_weight: 586739
      trx_mysql_thread_id: 2
      trx_query: DELETE FROM
employees.salaries WHERE salary > 65000
      trx_operation_state: updating or deleting
      trx_tables_in_use: 1
      trx_tables_locked: 1
      trx_lock_structs: 3003
      trx_lock_memory_bytes: 450768
      trx_rows_locked: 1407513
      trx_rows_modified: 583736
      trx_concurrency_tickets: 0
      trx_isolation_level: REPEATABLE READ
      trx_unique_checks: 1
      trx_foreign_key_checks: 1
      trx_last_foreign_key_error: NULL
      trx_adaptive_hash_latched: 0
      trx_adaptive_hash_timeout: 10000
      trx_is_read_only: 0
      trx_autocommit_non_locking: 0
```

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In addition to the information shown in the slide, INNODB_TRX contains the transaction's isolation level, the approximate number of rows and tables locked, and whether this is a read-only transaction.

As with other Information Schema views, the INNODB_TRX view contains current information but does not contain any historical information. This means that you cannot use it to examine queries that are no longer blocked, or queries that have completed.

The Performance Schema data_locks Table

Shows data locks held and requested

```
mysql> SELECT * FROM data_locks\G
***** 1. row *****
    ENGINE: INNODB
    ENGINE_LOCK_ID: 4140:74
    ENGINE_TRANSACTION_ID: 4140
    THREAD_ID: 37
    EVENT_ID: 9
    OBJECT_SCHEMA: test
    OBJECT_NAME: t1
    PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
    INDEX_NAME: NULL
    OBJECT_INSTANCE_BEGIN: 140489308280888
    LOCK_TYPE: TABLE
    LOCK_MODE: IX
    LOCK_STATUS: GRANTED
    LOCK_DATA: NULL
***** 2. row *****
    ENGINE: INNODB
    ENGINE_LOCK_ID: 4140:66:5:1
    ENGINE_TRANSACTION_ID: 4140
    THREAD_ID: 37
    EVENT_ID: 9
    OBJECT_SCHEMA: test
    OBJECT_NAME: t1
    PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
    INDEX_NAME: GEN_CLUST_INDEX
    OBJECT_INSTANCE_BEGIN: 140489320307736
    LOCK_TYPE: RECORD
    LOCK_MODE: X
    LOCK_STATUS: GRANTED
    LOCK_DATA: supremum pseudo-record
```

Identifies the lock

Describes the lock

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

ENGINE: The storage engine that holds or requested the lock

ENGINE_LOCK_ID: The ID of the lock held or requested by the storage engine

ENGINE_TRANSACTION_ID: The storage engine internal ID of the transaction that requested the lock. To obtain details about the transaction, join this column with the `TRX_ID` column of the `INFORMATION_SCHEMA_INNODB_TRX` table.

THREAD_ID: The ID of the thread that owns the lock. To obtain details about the thread, join this column with the `THREAD_ID` column of the Performance Schema threads table.

EVENT_ID: The Performance Schema event that caused the lock. The `THREAD_ID` and `EVENT_ID` values implicitly identify a parent event in other Performance Schema tables:

- The parent wait event in `events_waits_xxx` tables
- The parent stage event in `events_stages_xxx` tables
- The parent statement event in `events_statements_xxx` tables
- The parent transaction event in `events_transactions_xxx` tables

To obtain details about the parent event, join the `THREAD_ID` and `EVENT_ID` columns with the columns of the same name in the appropriate parent event table.

OBJECT_SCHEMA: The database that contains the locked table

OBJECT_NAME: The name of the locked table

INDEX_NAME: The name of the locked index, if any; `NULL` otherwise. InnoDB always creates an index (`GEN_CLUST_INDEX`), so `INDEX_NAME` is non-`NULL` for InnoDB tables.

OBJECT_INSTANCE_BEGIN: The address in memory of the lock

LOCK_TYPE: The type of lock. For InnoDB, permitted values are RECORD for a row-level lock, TABLE for a table-level lock.

LOCK_MODE: How the lock is requested

LOCK_STATUS: The status of the lock request. For InnoDB, permitted values are GRANTED (lock is held) and PENDING (lock is being waited for).

LOCK_DATA: The data associated with the lock, if any. The value is storage engine dependent. For InnoDB, values are primary key values of the locked record if **LOCK_TYPE** is RECORD; otherwise NULL. This column contains the values of the primary key columns in the locked row, formatted as a valid SQL string. If there is no primary key, **LOCK_DATA** is the unique InnoDB internal row ID number. If a gap lock is taken for key values or ranges above the largest value in the index, **LOCK_DATA** reports supremum pseudo-record. When the page containing the locked record is not in the buffer pool (because it was paged out to disk while the lock was held), InnoDB does not fetch the page from disk, to avoid unnecessary disk operations. Instead, **LOCK_DATA** is set to NULL.

The Performance Schema data_lock_waits Table

Shows which held data locks are blocking which requested data locks

```
mysql> SELECT * FROM data_lock_waits\G
***** 1. row *****
    ENGINE: INNODB
REQUESTING_ENGINE_LOCK_ID: 4141:66:5:2
REQUESTING_ENGINE_TRANSACTION_ID: 4141
    REQUESTING_THREAD_ID: 38
    REQUESTING_EVENT_ID: 11
REQUESTING_OBJECT_INSTANCE_BEGIN: 140489320441368
BLOCKING_ENGINE_LOCK_ID: 4140:66:5:2
BLOCKING_ENGINE_TRANSACTION_ID: 4140
    BLOCKING_THREAD_ID: 37
    BLOCKING_EVENT_ID: 9
BLOCKING_OBJECT_INSTANCE_BEGIN: 140489320307736
```

What is requesting the lock

What is blocking the lock



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

REQUESTING_ENGINE_LOCK_ID: The ID of the lock requested by the storage engine. To obtain details about the lock, join this column with the `ENGINE_LOCK_ID` column of the `data_locks` table.

REQUESTING_ENGINE_TRANSACTION_ID: The storage engine internal ID of the transaction that requested the lock

REQUESTING_THREAD_ID: The thread ID of the session that requested the lock

REQUESTING_EVENT_ID: The Performance Schema event that caused the lock request in the session that requested the lock

BLOCKING_ENGINE_LOCK_ID: The ID of the blocking lock. To obtain details about the lock, join this column with the `ENGINE_LOCK_ID` column of the `data_locks` table.

BLOCKING_ENGINE_TRANSACTION_ID: The storage engine internal ID of the transaction that holds the blocking lock

BLOCKING_THREAD_ID: The thread ID of the session that holds the blocking lock

BLOCKING_EVENT_ID: The Performance Schema event that caused the blocking lock in the session that holds it

sys.innodb_lock_waits View

A simpler approach to troubleshoot locking issues is to use the `sys.innodb_lock_waits` view to query blocked (waiting) and blocking statements.

- Combines information from multiple views:
 - `INFORMATION_SCHEMA.INNODB_TRX`
 - `performance_schema.data_locks`
 - `performance_schema.data_lock_waits`
- Links blocking to waiting transactions and information about each transactions



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

sys.innodb_lock_waits: Example Query

```
mysql> SELECT * FROM innodb_lock_waits\G
***** 1. row *****
    wait_started: date-and-time
        wait_age: 00:00:22
    wait_age_secs: 22
    locked_table: `sbtest`.`sbtest1`
locked_table_schema: sbtest
locked_table_name: sbtest1
locked_table_partition: NULL
locked_table_subpartition: NULL
    locked_index: PRIMARY
    locked_type: RECORD
    waiting_trx_id: 17000
    waiting trx started: date-and-time
    waiting trx age: 00:00:22
    waiting trx rows locked: 1
    waiting trx rows modified: 0
    waiting pid: 14
    waiting query: update sbtest1 set k
= k + 1 where id = 14
    waiting lock id: 17000:195:5:15
    waiting lock mode: X
                                blocking trx id: 16999
                                blocking pid: 12
                                blocking query: NULL
                                blocking lock id: 16999:195:5:15
                                blocking lock mode: X
blocking trx started: date-and-time
blocking trx age: 00:00:27
blocking trx rows locked: 1
blocking trx rows modified: 1
    sql kill blocking query: KILL QUERY 12
sql kill blocking connection: KILL 12
1 row in set (#.## sec)
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Metadata Locks

- InnoDB blocks DDL operations on a table if another transaction is accessing it, by using metadata locks.
- Monitor these locks by querying the Performance Schema `metadata_locks` table.

```
mysql> SELECT * FROM
performance_schema.metadata_locks\G
***** 1. row ***** 2. row *****
OBJECT_TYPE: USER LEVEL LOCK          OBJECT_TYPE: USER LEVEL LOCK
OBJECT_SCHEMA: NULL                    OBJECT_SCHEMA: NULL
OBJECT_NAME: test1                   OBJECT_NAME: test1
OBJECT_INSTANCE_BEGIN: 139783346259664 OBJECT_INSTANCE_BEGIN: 139733113368448
LOCK_TYPE: EXCLUSIVE                 LOCK_TYPE: EXCLUSIVE
LOCK_DURATION: EXPLICIT              LOCK_DURATION: EXPLICIT
LOCK_STATUS: GRANTED                LOCK_STATUS: PENDING
SOURCE: item_func.cc:5532           SOURCE: item_func.cc:5532
OWNER_THREAD_ID: 33                  OWNER_THREAD_ID: 34
OWNER_EVENT_ID: 3
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Reading the metadata_locks Table

Use the `LOCK_STATUS` column to indicate the status of each lock:

- When a metadata lock is requested and obtained immediately, MySQL inserts a new row with a status of **GRANTED**.
- When a metadata lock is requested and not obtained immediately, MySQL inserts a new row with a status of **PENDING**.
 - When the metadata lock is obtained, its row status is updated to **GRANTED**.
- When a metadata lock is released, its row is deleted.
- When a pending lock request is canceled by the deadlock detector to break a deadlock (`ER_LOCK_DEADLOCK`), its row status is updated from **PENDING** to **VICTIM**.
- When a pending lock request times out (`ER_LOCK_WAIT_TIMEOUT`), its row status is updated from **PENDING** to **TIMEOUT**.
- When a granted lock or pending lock request is killed, its row status is updated from **GRANTED** or **PENDING** to **KILLED**.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The **VICTIM**, **TIMEOUT**, and **KILLED** status values are short-lived and signify that the lock row is about to be deleted.

You might also see the `PRE_ACQUIRE_NOTIFY` and `POST_RELEASE_NOTIFY` status values. These are short-lived and signify that the metadata locking subsystem is notifying interested storage engines while entering lock acquisition or leaving lock release operations.

Topics

- Stability
- Why Databases Fail
- Capacity Planning
- Troubleshooting
- Identifying the Causes of Server Slowdowns
- InnoDB Recovery



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

InnoDB Recovery

- InnoDB recovers automatically after a failure.
- Use `CHECK TABLE` or a client program to find inconsistencies, incompatibilities, and other problems.
 - InnoDB automatically detects problems with stored data when you access it.
 - `CHECK TABLE` forces InnoDB to access all data.
- Restore a table by dumping it with `mysqldump`, dropping it, and re-creating it from the dump file.
- To repair tables after a crash, restart the server with the `--innodb_force_recovery` option, or restore tables from a backup.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

`mysqldump` is covered in the lesson titled “Performing Backups.”

Using --innodb_force_recovery

If InnoDB automatic recovery fails, use the following procedure:

1. Restart the server with `--innodb_force_recovery=1`, and increase it until InnoDB starts.
 - The option accepts a value from 0 through 6.
 - 0 is the default, indicating default automatic recovery.
 - Take extra care when using larger values. These prevent INSERT, UPDATE, or DELETE operations and might result in inconsistencies in the recovered tables.
 - 4 and above place InnoDB in read-only mode.
2. Dump the InnoDB tables and then drop them while the option is in effect.
3. Restart the server *without* the `--innodb_force_recovery` option. When the server comes up, recover the InnoDB tables from the dump files.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Always exercise caution when using `--innodb_force_recovery`. It is good practice to take a copy of the data directory before attempting any recovery.

Summary



In this lesson, you should have learned how to:

- Improve the stability of MySQL servers
- Monitor database growth and explain capacity planning
- Troubleshoot server slowdowns
- Identify locked resources
- Perform InnoDB crash recovery

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practices

- 8-1: Quiz – Maintaining a Stable System
- 8-2: Identifying the Cause of Slow Performance



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.