

MySQL for Database Administrators

Student Guide - Volume II

D61762GC51 | D108206

Learn more from Oracle University at education.oracle.com



Author

KimSeong Loh

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

Disclaimer

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

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

Restricted Rights Notice

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

U.S. GOVERNMENT RIGHTS

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

Trademark Notice

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

Editors

Moushmi Mukherjee
Aju Kumar

Graphic Designer

Kavya Bellur

Publishers

Sujatha Nagendra
Pavithran Adka
Syed Ali

5102272020

Contents

1 Introduction to MySQL

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

2 Installing and Upgrading MySQL

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

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

3 Understanding MySQL Architecture

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

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

4 Configuring MySQL

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

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

5 Monitoring MySQL

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

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

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

6 Managing MySQL Users

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

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

7 Securing MySQL

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

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

8 Maintaining a Stable System

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

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

9 Optimizing Query Performance

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

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

10 Choosing a Backup Strategy

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

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

11 Performing Backups

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

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

12 Configuring a Replication Topology

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

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

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

13 Administering a Replication Topology

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

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

14 Achieving High Availability with MySQL InnoDB Cluster

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

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

15 Conclusion

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

7

Securing MySQL



ORACLE®

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

Objectives



After completing this lesson, you should be able to:

- Recognize common security risks
- List security problems and countermeasures for networks, passwords, operating systems, filesystems, 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
- Encrypt InnoDB tablespaces and log files
- Configure and use MySQL Enterprise Firewall



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

Topics

- Security Risks

- Network Security
- Secure Connections
- Password Security
- Operating System Security
- Encrypting Data-at-Rest
- Protecting Against SQL Injections
- MySQL Enterprise Firewall



ORACLE®

Copyright © 2020, 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



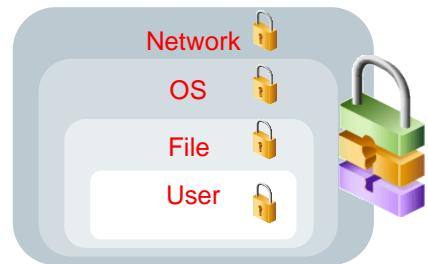
ORACLE®

Copyright © 2020, 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 with a weak password or without a password is vulnerable to attack.
- Operating system:
 - Extra user accounts on a server can increase the vulnerability of the MySQL installation.
- Filesystem:
 - Directories, database files, and log files on the server can be opened by users who should not have access.



ORACLE®

Copyright © 2020, 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
- Encrypting Data-at-Rest
- Protecting Against SQL Injections
- MySQL Enterprise Firewall



ORACLE®

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

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

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. If external access is authorized by the firewall, it passes the request to the MySQL server. Responses from the MySQL server pass back out to the external access.

A demilitarized zone (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.

Preventing Network Security Risks

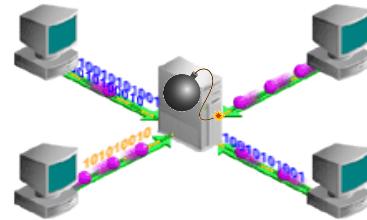
- Use a firewall to control access to MySQL ports.
- Consider limiting the network interfaces used by the server.
- Ensure that MySQL accounts are protected with strong passwords and do not have unnecessary privileges.
- Do not grant more permissions than a user requires (principle of least privilege).
- Ensure that only authorized clients can connect to the server to access its databases.
- Do not transmit plain (unencrypted) data over networks.



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

Topics

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



ORACLE®

Copyright © 2020, 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 identity, 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 © 2020, 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, TLSv1.2, and TLSv1.3, with TLSv1.3 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.

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



Copyright © 2020, 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 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 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.



ORACLE®

Copyright © 2020, 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 © 2020, 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
```

- Connections through UNIX socket or named pipes do not use SSL.



Copyright © 2020, 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.

The DBA can also use the Performance Schema to verify that connections use SSL, for example:

```
mysql> SELECT * FROM sys.session_ssl_status\G
***** 1. row *****
    thread_id: 50
    ssl_version: TLSv1.2
    ssl_cipher: DHE-RSA-AES128-GCM-SHA256
ssl_sessions_reused: 0
***** 2. row *****
    thread_id: 51
    ssl_version:
    ssl_cipher:
ssl_sessions_reused: 0
2 row in set (0.00 sec)
```

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 © 2020, 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



ORACLE®

Copyright © 2020, 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 © 2020, 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.
 - The default value of `tls_version` is the list of all protocols supported by the SSL library that is used to compile MySQL.
- 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 © 2020, 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 © 2020, 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 such as 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 © 2020, 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 © 2020, 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 © 2020, 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 © 2020, 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 © 2020, 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



ORACLE

Copyright © 2020, 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.



ORACLE®

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

The following is an 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 © 2020, 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 © 2020, 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



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

Answer: c

Topics

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



ORACLE®

Copyright © 2020, 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 non-administrative users from reading this table.
 - Encrypt the `mysql` tablespace that stores the data dictionary.



ORACLE®

Copyright © 2020, 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 © 2020, 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 Component

- This component is 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';
```
- Loads the component
  - Registers it in the `mysql.component` system table so that it loads automatically when the server is restarted
- Uninstall the component by using the `UNINSTALL COMPONENT` statement:

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



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

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

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.

## Validate Password Component Variables

- 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/lowercase, special characters
  - **2 or STRONG:** Length, numeric, upper/lowercase, 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.
  - 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 **ON** by default; it rejects a password that is the same as the username or its reverse.
  - This option is not affected by the setting of the policy variable.



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

When you install the Validate Password component, you can access the variables prefixed with “`validate_password`.” If the password validation component is not installed, the variables do not exist.

The following output shows the default values of these 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 (0.00 sec)
```

## 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
```

- Use SET PERSIST

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



Copyright © 2020, 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 © 2020, 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 © 2020, 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 username 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`
  - Can be changed using the `default_authentication_plugin` system variable
- If you require other authentication methods that store their credentials somewhere other than the `mysql.user` table, then 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 © 2020, 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 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 SHA2 () functions and store the password's hash value.
  - Alternatively, use some other one-way hashing function available to the application.



ORACLE®

Copyright © 2020, 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 © 2020, 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.

Conditions on setting the delay:

- `connection_control_min_connection_delay` cannot be set greater than the current value of `connection_control_max_connection_delay`.
- `connection_control_max_connection_delay` cannot be set less than the current value of `connection_control_min_connection_delay`.

Due to these conditions, you may have to set the delay in a specific order.

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

## Using the CONNECTION\_CONTROL\_FAILED\_LOGIN\_ATTEMPTS Plugin

- Install the plugin by 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 © 2020, 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.



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

**Answer: b**

## Topics

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



ORACLE®

Copyright © 2020, 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.



ORACLE®

Copyright © 2020, 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 © 2020, 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.
- Put MySQL behind the firewall or in a demilitarized zone (DMZ).



ORACLE®

Copyright © 2020, 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).



ORACLE®

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

## Topics

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



ORACLE®

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

## Keyring

- Enables MySQL Server internal components and plugins to securely store sensitive information for later retrieval
- Is implemented as a plugin
  - Different types of keyring plugins are available in the Community and Enterprise Edition of MySQL.
- Is used to store:
  - InnoDB storage engine encryption master key
  - Audit log file encryption password
  - Binary log encryption master key
- Includes a SQL interface for keyring key management
  - Consists of a set of general-purpose user-defined functions (UDFs) installed from the `keyring_udf` plugin
  - Allows application users to store their own keys securely in the keyring



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

This is a list of keyring plugins available in the MySQL Enterprise Edition:

- `keyring_file`: Stores keyring data in a file on the server host
- `keyring_encrypted_file`: Stores keyring data in an encrypted file on the server host
- `keyring_okv`: Can be used with KMIP-compatible back-end keyring storage products such as Oracle Key Vault and Gemalto SafeNet KeySecure Appliance
- `keyring_aws`: Communicates with the Amazon Web Services Key Management Service for key generation and uses a local file for key storage
- `keyring_hashicorp`: Communicates with HashiCorp Vault for back-end storage (available in MySQL 8.0.18 and later)

The `keyring_file` plugin is available in the MySQL Community Edition.

When the data-at-rest encryption feature uses a centralized key management solution (such as `keyring_okv` and `keyring_hashicorp`), the feature is referred to as “MySQL Enterprise Transparent Data Encryption (TDE).”

## Deploying a Keyring

- The keyring plugin must be loaded early during the server startup sequence.
  - Use the `--early-plugin-load` option.
  - Other server components may need to access it during their own initialization.
    - For example, InnoDB needs the master key to decrypt the redo and undo logs during startup.
- Only one keyring plugin should be enabled at a time.
- Example of enabling the encrypted file keyring:

```
[mysqld]
early-plugin-load=keyring_encrypted_file.so
keyring_encrypted_file_data=/var/lib/mysql-keyring/keyring-encrypted
keyring_encrypted_file_password=mysecret
```

- Each keyring plugin has its own set of server variables to configure the keyring.



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

Refer to the following sections of the *MySQL Reference Manual* for the installation and configuration instructions of the keyring plugin that you want to deploy:

- For `keyring_file`:
  - <https://dev.mysql.com/doc/refman/8.0/en/keyring-file-plugin.html>
- For `keyring_encrypted_file`:
  - <https://dev.mysql.com/doc/refman/8.0/en/keyring-encrypted-file-plugin.html>
- For `keyring_okv`:
  - <https://dev.mysql.com/doc/refman/8.0/en/keyring-okv-plugin.html>
- For `keyring_aws`:
  - <https://dev.mysql.com/doc/refman/8.0/en/keyring-aws-plugin.html>
- For `keyring_hashicorp`:
  - <https://dev.mysql.com/doc/refman/8.0/en/keyring-hashicorp-plugin.html>

## Key Management Functions

- `keyring_key_generate(key_id, key_type, key_length)`
  - Generates a new random key with a given ID, type, and length, and stores it in the keyring
- `keyring_key_store(key_id, key_type, key)`
  - Obfuscates and stores a key in the keyring
- `keyring_key_fetch(key_id)`
  - Given a key ID, deobfuscates and returns the key value
- `keyring_key_type_fetch(key_id)`
  - Given a key ID, returns the key type
- `keyring_key_length_fetch(key_id)`
  - Given a key ID, returns the key length
- `keyring_key_remove(key_id)`
  - Removes the key with a given ID from the keyring



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

A key stored in the keyring by a given user can be manipulated later only by the same user. That is, the value of the `CURRENT_USER()` function at the time of key manipulation must have the same value as when the key was stored in the keyring.

Install the `keyring_udf` plugin and the UDFs:

```
INSTALL PLUGIN keyring_udf SONAME 'keyring_udf.so';
CREATE FUNCTION keyring_key_generate RETURNS INTEGER
 SONAME 'keyring_udf.so';
CREATE FUNCTION keyring_key_fetch RETURNS STRING
 SONAME 'keyring_udf.so';
CREATE FUNCTION keyring_key_length_fetch RETURNS INTEGER
 SONAME 'keyring_udf.so';
CREATE FUNCTION keyring_key_type_fetch RETURNS STRING
 SONAME 'keyring_udf.so';
CREATE FUNCTION keyring_key_store RETURNS INTEGER
 SONAME 'keyring_udf.so';
CREATE FUNCTION keyring_key_remove RETURNS INTEGER
 SONAME 'keyring_udf.so';
```

## Encrypting InnoDB Tablespaces

The following InnoDB tablespaces can be encrypted:

- File-per-table tablespace:

- Add the ENCRYPTION = 'Y' option to the CREATE TABLE or ALTER TABLE statement.

```
ALTER TABLE t1 ENCRYPTION = 'Y';
```

- General tablespace:

- Add the ENCRYPTION = 'Y' option to the CREATE TABLESPACE or ALTER TABLESPACE statement.

```
ALTER TABLESPACE ts1 ENCRYPTION = 'Y';
```

- mysql system tablespace:

- Use the ENCRYPTION = 'Y' option with the ALTER TABLESPACE statement.

```
ALTER TABLESPACE mysql ENCRYPTION = 'Y';
```



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

The data including indexes stored in the tablespace is encrypted when it is written to disk and decrypted when it is read from disk. The data cached in memory is stored in unencrypted format.

## Encrypting InnoDB Redo Logs and Undo Logs

### Redo log encryption:

- Is enabled by setting `innodb_redo_log_encrypt=ON`
- Affects new redo log pages written to disk
- Does not affect redo log pages already on disk
- Stores the tablespace encryption key in the header of the first redo log file
- Causes server restart to fail when the keyring plugin or master key is not available

### Undo log encryption:

- Is enabled by setting `innodb_undo_log_encrypt=ON`
- Affects new undo log pages written to disk
- Does not affect undo log pages already on disk
- Stores the tablespace encryption key in the header of the undo log file
- When disabled, the server continues to require the keyring plugin and master key until the undo tablespaces that contained the encrypted undo log data are truncated



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

As with tablespace data, redo log and undo log encryption occurs when the data is written to disk, and decryption occurs when the data is read from disk.

## InnoDB Encryption Keys

- InnoDB uses a two-tier encryption key architecture, consisting of a master encryption key and tablespace keys.
- A tablespace key is used to encrypt and decrypt the contents of a tablespace file, including the redo log files and undo tablespaces.
  - The tablespace key is encrypted using the master key and stored in the tablespace header.
  - The master key is stored in the keyring.
- The master key can be rotated by using the command:

```
ALTER INSTANCE ROTATE INNODB MASTER KEY
```

- The user requires the `ENCRYPTION_KEY_ADMIN` or `SUPER` privilege.
- All tablespace keys in the MySQL Server are re-encrypted and saved back to their respective tablespace headers.
- This operation is atomic.
- It does not decrypt or re-encrypt the associated tablespace data.



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

## Encrypting Binary Log and Relay Log

- From MySQL 8.0.14, binary log files and relay log files can be encrypted.
  - Enable the encryption by setting the `binlog_encryption` system variable to `ON`.
- When the `binlog_encryption` system variable is changed during run time:
  - The binary log and relay log files are rotated.
  - New log files follow the new setting.
  - Old log files are not affected and remain as they are.
- The `SHOW BINARY LOGS` statement shows whether each binary log file is encrypted or unencrypted.
- `mysqlbinlog` cannot read encrypted binary log files directly.
  - Use the `--read-from-remote-server` option to connect to a running MySQL Server to read the encrypted binary logs.



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

## Binary Log Encryption Keys

- A randomly generated 32-byte **file password** is used to encrypt the contents of each binary log file and relay log file.
- The file password is then encrypted using the binary log master key.
  - The master key is stored in the keyring.
  - The encrypted file password is stored in the log file's header.
- The master key can be rotated using the command:

```
ALTER INSTANCE ROTATE BINLOG MASTER KEY
```

  - A new binary log master key is generated.
  - All the file passwords of known log files are re-encrypted using the new master key.
    - Uses the binary log index and relay log index files to identify the log files
  - The content of the log files does not need to be re-encrypted.
- If you want the binary log master key to be rotated whenever the server restarts, set the `binlog_rotate_encryption_master_key_at_startup` system variable to ON.



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

## Topics

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



ORACLE®

Copyright © 2020, 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.



ORACLE®

Copyright © 2020, 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 that 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 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 © 2020, 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.
- If these characters are required, the application can be programmed to escape them to remove the special meaning so that they will be treated as part of a string literal.



Copyright © 2020, 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.



ORACLE®

Copyright © 2020, 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
- Encrypting Data-at-Rest
- Protecting Against SQL Injections
- MySQL Enterprise Firewall



ORACLE®

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

## MySQL Enterprise Firewall

- Plugin available with Enterprise Edition only
- Is an 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 © 2020, 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



ORACLE®

Copyright © 2020, 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()`



ORACLE

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

MySQL Enterprise Firewall maintains account and whitelist information. It uses `INFORMATION_SCHEMA` tables to provide views into cached data, and tables in the `mysql` system database to store this data in persistent form. The `INFORMATION_SCHEMA` tables are accessible by anyone. The `mysql` tables can be accessed only by users with privileges for that database.

The `INFORMATION_SCHEMA.MYSQL_FIREWALL_USERS` and `mysql.firewall_users` tables list registered firewall accounts and their operational modes.

The `INFORMATION_SCHEMA.MYSQL_FIREWALL_WHITELIST` and `mysql.firewall_whitelist` tables list registered firewall accounts and their whitelists.

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

## Enabling Firewall Protection

- Set the account to **PROTECTING** mode.

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

- 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.

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

- The server writes a message to the error log.

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



Copyright © 2020, 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**.
  - It is set to **ON** by default.



Copyright © 2020, 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 (#.## secs)
```



Copyright © 2020, 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



Copyright © 2020, 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 countermeasures 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
- Encrypt InnoDB tablespaces and log files
- Configure and use MySQL Enterprise Firewall



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

## Practices

- 7-1: Quiz—Securing MySQL
- 7-2: Enabling SSL for Secure Connections
- 7-3: Encrypting MySQL Data-at-Rest
- 7-4: Configuring MySQL Enterprise Firewall



ORACLE®

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



8

# Maintaining a Stable System



ORACLE®

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

## 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



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

## Topics

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



ORACLE®

Copyright © 2020, 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.



ORACLE®

Copyright © 2020, 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 the 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.



ORACLE®

Copyright © 2020, 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: filesystem, memory, and CPU usage
  - `top, iostat, vmstat, sysstat, sar` on Linux- or UNIX-based systems
  - Resource Monitor and Performance Monitor on Windows
- Record MySQL status and configuration:
  - `SHOW PROCESSLIST` or `sys.session` to see the running processes
  - `mysqladmin extended-status` to see status variables
    - Use `--relative` to record value deltas.
- Profile application use-case response times:
  - Log in, search, create, read, update, and delete.



Copyright © 2020, 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 © 2020, 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
- Locking
- InnoDB Recovery



ORACLE®

Copyright © 2020, 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 balancer
  - Application software
    - User-facing components
    - Framework stability
- Consider *force majeure*:
  - Natural disasters or other extraordinary events

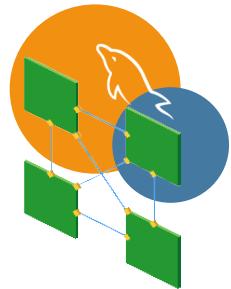


ORACLE®

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

## Server Hardware

- The server room is an important part of a database environment.
  - Ensure that 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 © 2020, 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 activities 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 © 2020, 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 © 2020, 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 that the filesystem does not fill up the disks with MySQL data or logs, or logs from other services.



Copyright © 2020, 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 MySQL 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



ORACLE®

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

## Topics

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



ORACLE®

Copyright © 2020, 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 © 2020, 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, and header and footer information



ORACLE®

Copyright © 2020, 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 © 2020, 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.

**Note:** DATA\_LENGTH and INDEX\_LENGTH values are only estimates for InnoDB tables. The values are storage engine dependent.

## 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)
```

- The value of the TOTAL\_EXTENTS column will be 0 until the first extent has been filled.
  - The departments and dept\_manager tables each contain less than EXTENT\_SIZE (1 MB by default).



Copyright © 2020, 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.
- General tablespaces and the system tablespace store multiple tables in each file.
  - The file size is the combined size of all the tables stored in the tablespace.
  - The physical size of an individual table cannot be determined.

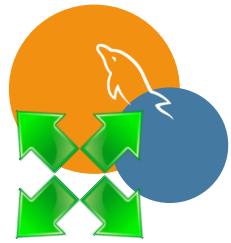


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## 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



ORACLE®

Copyright © 2020, 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 © 2020, 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 read operations?

- 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



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

**Answer: b**

## Topics

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



ORACLE®

Copyright © 2020, 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
    - Filesystem 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 © 2020, 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 © 2020, 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 © 2020, 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
  - Optimize the queries.
  - Fine-tune database settings.



Copyright © 2020, 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
- Locking
- InnoDB Recovery



ORACLE®

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

## Quiz



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.



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

**Answer: c**

## Topics

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



ORACLE®

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

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

## Identifying Lock Contention

- Understand how InnoDB locks rows.
- Identify long-running or blocked queries by using `SHOW PROCESSLIST` or by querying the Performance Schema `threads` table.
- 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 © 2020, 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 |



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

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

InnoDB can perform nonlocking read. A normal `SELECT` statement without the `FOR SHARE` or `FOR UPDATE` clause can read InnoDB data without any locking. `FOR SHARE` will require a shared lock and `FOR UPDATE` will require an exclusive lock.

## Troubleshooting Locks by Using 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.

- Querying the Performance Schema threads table can provide the same information. This is better than SHOW PROCESSLIST because:

- No mutex lock required
- Less impact to server performance



Copyright © 2020, 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.

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

## 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 © 2020, 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.

## 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



Copyright © 2020, 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.

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

## Performance Schema metadata\_locks Table

- 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 *****
OBJECT_TYPE: TABLE
OBJECT_SCHEMA: world
OBJECT_NAME: city
COLUMN_NAME: NULL
OBJECT_INSTANCE_BEGIN: 140226189198448
LOCK_TYPE: SHARED_READ_ONLY
LOCK_DURATION: TRANSACTION
LOCK_STATUS: GRANTED
SOURCE: sql_parse.cc:6014
OWNER_THREAD_ID: 48
OWNER_EVENT_ID: 132
***** 2. row *****
OBJECT_TYPE: TABLE
OBJECT_SCHEMA: world
OBJECT_NAME: city
COLUMN_NAME: NULL
OBJECT_INSTANCE_BEGIN: 140226254200240
LOCK_TYPE: EXCLUSIVE
LOCK_DURATION: TRANSACTION
LOCK_STATUS: PENDING
SOURCE: sql_parse.cc:6014
OWNER_THREAD_ID: 49
OWNER_EVENT_ID: 16
```



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

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**.

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.

## sys.schema\_table\_lock\_waits View

- Display which sessions are blocked waiting on metadata locks, and what is blocking them

```
mysql> select * from schema_table_lock_waits\G
***** 1. row *****
 object_schema: world
 object_name: city
 waiting_thread_id: 49
 waiting_pid: 10
 waiting_account: root@localhost
 waiting_lock_type: EXCLUSIVE
 waiting_lock_duration: TRANSACTION
 waiting_query: TRUNCATE world.city
 waiting_query_secs: 243
 waiting_query_rows_affected: 0
 waiting_query_rows_examined: 0
 blocking_thread_id: 48
 blocking_pid: 9
 blocking_account: root@localhost
 blocking_lock_type: SHARED_READ_ONLY
 blocking_lock_duration: TRANSACTION
 sql_kill_blocking_query: KILL QUERY 9
 sql_kill_blocking_connection: KILL 9
```



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

## Topics

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



ORACLE®

Copyright © 2020, 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 by using the `--innodb_force_recovery` option, or restore tables from a backup.
  - It is recommended to back up the database before repairing the tables.



ORACLE

Copyright © 2020, 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. Shut down the server and back up all the data files.
2. 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.
3. Dump the InnoDB tables and then drop them while the option is in effect.
4. Restart the server *without* the `--innodb_force_recovery` option. When the server comes up, recover the InnoDB tables from the dump files.



Copyright © 2020, 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



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

## Practices

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



ORACLE®

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

9

# Optimizing Query Performance



ORACLE®

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

## Objectives



After completing this lesson, you should be able to:

- Identify queries that are suitable for optimization
- Examine query index usage
- Create indexes to improve server performance
- Maintain index statistics
- Use MySQL Query Analyzer



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

## Topics

- Identifying slow queries
  - EXPLAIN statement
  - Working with indexes
  - Index statistics
  - MySQL Query Analyzer

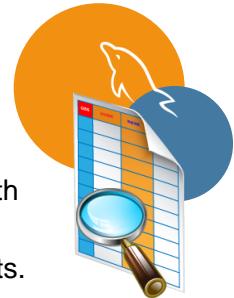


ORACLE®

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

## Identifying Slow Queries

- Identify the queries that the server executes:
  - Use the general query log and slow query log.
- Identify the queries that take a long time:
  - Use the slow query log.
  - Prioritize these over outlier queries or queries that you execute infrequently.
- Find queries that execute many times:
  - Regularly execute **SHOW PROCESSLIST** or query the Performance Schema threads table to see currently executing statements and their durations and to identify emerging patterns.
  - Use **sys.createStatement\_analysis** to view normalized statements with aggregated statistics.
  - Use the slow query log with a low threshold to record most statements.



ORACLE®

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

## Choosing What to Optimize

- Do not assume that you need to optimize only the slowest queries.
  - You might improve the overall performance of the system by optimizing the most frequently executed queries, even if such queries are already executing relatively quickly.
    - Optimizing a frequent query (even by only a small amount) can free up resources, reduce locking, and improve the response time and throughput of the server.
    - Optimizing an infrequent query provides less overall benefit.
- Consider the following examples:
  - You optimize a query that takes two seconds so that it now takes 800 ms, but this query executes on average once per minute.
  - You optimize a query that takes 20 ms so that it now takes 15 ms, but this query executes several thousand times per minute.



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

If the server spends an extra 5 ms on a query that executes several thousand times per minute, that might translate to 15 or 20 seconds of cumulatively delayed statement execution on that server. Small optimizations on frequent queries might give more overall improvement.

## Topics

- Identifying slow queries
- EXPLAIN statement
- Working with indexes
- Index statistics
- MySQL Query Analyzer



ORACLE®

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

## Using EXPLAIN to See Optimizer's Choice of Index

- Produces a query execution plan showing how the optimizer plans to execute a given SQL statement
  - Includes information from MySQL server optimizations
- Examines SELECT, INSERT, REPLACE, UPDATE, and DELETE statements
- Does not return any data from the data sets; does not perform any data modification in the statement
- Chooses optimal operations based on:
  - Query
  - Structure of tables in the query
  - Any relevant indexes



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

## EXPLAIN: Example

```
mysql> EXPLAIN SELECT * FROM employees WHERE emp_no=10001\G
***** 1. row *****
 id: 1
 select_type: SIMPLE
 table: employees
 partitions: NULL
 type: const
possible_keys: PRIMARY
 key: PRIMARY
 key_len: 4
 ref: const
 rows: 1
 filtered: 100.00
 Extra: NULL
1 row in set, 1 warning (#.## sec)
```



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

The output in the slide generates a warning. When you use EXPLAIN to view the plan for a SELECT statement, it produces a note-level event that describes query rewriting and optimization actions.

When the `sql_notes` system variable is set to 1 (the default), note-level events are treated as warnings. For more information, see the section titled “Displaying Query Rewriting and Optimization Actions” later in this lesson.

## EXPLAIN Output

- **id**: Number identifying the examined statement
- **select\_type**: Type of select used in the query
  - SIMPLE: The query does not use UNION or subqueries.
  - Other values indicate the type of union or subquery.
  - For DML statement: INSERT, REPLACE, UPDATE, and DELETE
- **table**: Table for the output row
- **partitions**: Partitions that the optimizer needs to examine to execute the query
- **type**: Index or join comparison type
- **possible\_keys**: Indexes that are relevant to the query
- **key**: Specific index chosen by the optimizer



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

## EXPLAIN Output

- **key\_len**: Size (in bytes) of the left-most columns used to search the index
- **ref**: Columns (or `const`) compared to the index
- **rows**: Estimated number of rows that the optimizer predicts will be returned by the query
- **filtered**: Percentage of rows that are filtered by the table condition
- **Extra**: Additional per-query information provided by the optimizer or storage engine



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

## Common type Values

The `type` column indicates the type of comparison used by the optimizer to access rows.

- **ALL**: Full table scan
- **index**: Full index scan
- **const**: Matching a primary or unique key against a constant at the start of a query
- **eq\_ref**: Matching a single referenced value (identified by the `ref` column) for equality
- **ref**: Matching one or more referenced values for equality
- **range**: Matching rows in a range that the named index (key) supports



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

See <https://dev.mysql.com/doc/refman/8.0/en/explain-output.html#explain-join-types> for the full list of type values.

## Displaying Query Rewriting and Optimization Actions

- Use `SHOW WARNINGS` to display further details about optimizations.
- Each displayed message provides extended information about the optimizer's plan for the query.
- It shows a rephrased version of the query in pseudo-SQL that represents the optimized sequence of operations, for example:

```
mysql> SHOW WARNINGS\G

1. row ****
Level: Note
Code: 1003
Message: /* select#1 */ select '10001' AS `emp_no`, '1953-09-02' AS
`birth_date`, 'Georgi' AS `first_name`, 'Facello' AS `last_name`, 'M' AS
`gender`, '1986-06-26' AS `hire_date` from `employees`.`employees` where 1
1 row in set (#.## sec)
```

- `SHOW WARNINGS` also shows any messages that are specific to the storage engine's own optimizations, if any.



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

`EXPLAIN` produces note-level messages that have the error code 1003. That error code has the internal symbol `ER_YES` and the associated error name `YES`. When `sql_notes` is set to 1, note-level messages are treated as warnings. Although they are not warnings or errors in the usual sense, they enable you to view these extra messages using the client as an interface.

Some other storage engines, such as NDB, add messages to the output of `SHOW WARNINGS` that indicate storage engine-specific optimizations.

## EXPLAIN Example: Table Scan

```
mysql> EXPLAIN SELECT *
-> FROM employees
-> WHERE first_name='Mong'\G
***** 1. row *****
 id: 1
select_type: SIMPLE
 table: employees
 partitions: NULL
 type: ALL
possible_keys: NULL
 key: NULL
key_len: NULL
 ref: NULL
 rows: 299246
filtered: 10.00
 Extra: Using where
1 row in set, 1 warning (0.00 sec)
```



- The `possible_keys`, `key`, and `key_len` columns display `NULL`.
  - The query cannot use any indexes to improve the query's performance.
- The `type` column shows the value `ALL`, indicating a table scan.
- The `rows` column shows the number 299246.
  - This is InnoDB's *estimate* of the number of rows in the table.

ORACLE

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

InnoDB generates an estimate of the number of rows in a table or an index when you run `ANALYZE TABLE tablename`.

## EXPLAIN Example: Primary Key

```
mysql> EXPLAIN SELECT *
-> FROM employees
-> WHERE emp_no=10001\G

 1. row ****
 id: 1
 select_type: SIMPLE
 table: employees
 partitions: NULL
 type: const
possible_keys: PRIMARY
 key: PRIMARY
 key_len: 4
 ref: const
 rows: 1
 filtered: 100.00
 Extra: NULL
1 row in set, 1 warning (#.## sec)
```



- The type of the operation is **const**.
  - **const** queries match literal (constant) values against primary or unique keys.
- The chosen index is shown as the key value **PRIMARY**, with a size of four bytes.
  - This index is compared for equality with a referred constant value, shown by **ref: const**.
  - Comparing the primary key with the specified value produces at most a single row.
- In addition, the optimizer estimates that it needs to examine only one row, shown by the value in the **rows** column.

ORACLE®

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

## EXPLAIN Example: Non-unique Index

```
mysql> EXPLAIN SELECT emp_no
-> FROM dept_manager
-> WHERE dept_no='d001'\G
***** 1. row *****
 id: 1
 select_type: SIMPLE
 table: dept_manager
 partitions: NULL
 type: ref
possible_keys: PRIMARY,emp_no,dept_no
 key: dept_no
 key_len: 16
 ref: const
 rows: 2
filtered: 100.00
Extra: Using index
1 row in set, 1 warning (#.# sec)
```



- The `type` of the operation is `ref`.
  - Compares the column value with a referred value (a `const`)
  - Matches the referenced value with a nonunique column
    - Might match multiple rows
- The optimizer estimates the number of rows to be two.
  - The actual value depends on the data.

The Extra text `Using index` indicates that the query does not result in an additional seek to read the actual row's data.

ORACLE

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

The structure of the `dept_manager` table is as follows:

```
CREATE TABLE dept_manager (
 dept_no CHAR(4) NOT NULL,
 emp_no INT NOT NULL,
 from_date DATE NOT NULL,
 to_date DATE NOT NULL,
 KEY (emp_no),
 KEY (dept_no),
 FOREIGN KEY (emp_no) REFERENCES employees (emp_no)
 ON DELETE CASCADE,
 FOREIGN KEY (dept_no) REFERENCES departments (dept_no)
 ON DELETE CASCADE,
 PRIMARY KEY (emp_no,dept_no)
);
```

All the secondary index in InnoDB tables also contain the primary key values.

The `dept_no` index also contain the values of `emp_no`. Therefore, the output shows a “Using index” in the `Extra` column.

## EXPLAIN and Complex Queries

- EXPLAIN produces multiple output rows when you analyze a complex query.
  - A complex query is a query that contains subqueries, UNION clause, or JOIN clause.
- Each row uses an `id` to uniquely identify the statement that it refers to.
  - If you have a SELECT statement with a subquery or UNION that has its own SELECT statement, you will have two rows, each with a different `id`.
    - Each row refers to table operations in a separate statement.
  - For a single SELECT statement that joins two tables, you get two rows with the same `id`.
    - Both rows refer to table operations within the same statement.



ORACLE®

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

## EXPLAIN Example: Simple Join

```
mysql> EXPLAIN SELECT first_name,
-> last_name, title
-> FROM employees
-> JOIN titles USING(emp_no)
-> WHERE title='Senior Engineer'\G
***** 1. row *****
 id: 1
 select_type: SIMPLE
 table: titles
 partitions: NULL
 type: index
possible_keys: PRIMARY,emp_no
 key: emp_no
 key_len: 4
 ref: NULL
 rows: 417756
 filtered: 10.00
Extra: Using where; Using index
***** 2. row *****
 id: 1
 select_type: SIMPLE
 table: employees
 partitions: NULL
 type: eq_ref
possible_keys: PRIMARY
 key: PRIMARY
 key_len: 4
 ref: employees.titles.emp_no
 rows: 1
 filtered: 100.00
 Extra: NULL
2 rows in set, 1 warning (#.## sec)
```



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

```
CREATE TABLE employees (
```

```
 emp_no INT NOT NULL,
 birth_date DATE NOT NULL,
 first_name VARCHAR(14) NOT NULL,
 last_name VARCHAR(16) NOT NULL,
 gender ENUM ('M', 'F') NOT NULL,
 hire_date DATE NOT NULL,
 PRIMARY KEY (emp_no)
);
```

```
CREATE TABLE titles (
```

```
 emp_no INT NOT NULL,
 title VARCHAR(50) NOT NULL,
 from_date DATE NOT NULL,
 to_date DATE,
 KEY (emp_no),
 FOREIGN KEY (emp_no) REFERENCES employees (emp_no) ON DELETE CASCADE,
 PRIMARY KEY (emp_no,title, from_date)
);
```

## Explanation of Simple Join Output

When you join tables in a query, the optimizer must perform an operation on each table.

- The join in the preceding slide has a `WHERE` clause that references an unindexed field.
- The first operation is an index scan of all rows in the `emp_no` index of the `titles` table.
  - It reads all the index entries, and filters them based on the `WHERE` clause against the primary key value. All secondary indexes in InnoDB tables also contain the primary key values.
- The second operation references the first one:
  - The type of the second operation is `eq_ref`, indicating that it matches each row for equality with its compared value.
  - The value in question is the qualified column name `employees.titles.emp_no`, indicating that the value to compare is not a constant provided in the statement, but comes from that other column.



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

## EXPLAIN FORMAT

- TRADITIONAL: (default)
  - Presents the output in the default tabular format
- JSON:
  - Presents the output in JSON format
  - Includes estimated cost and number of rows
- TREE:
  - Presents the output in a tree-like format
  - Includes estimated cost and number of rows
  - Is available as of MySQL 8.0.16

```
mysql> EXPLAIN FORMAT=TREE SELECT emp_no FROM dept_manager WHERE dept_no='d001'\G
***** 1. row *****
EXPLAIN: -> Filter: (dept_manager.dept_no = 'd001') (cost=0.45 rows=2)
-> Index lookup on dept_manager using dept_no (dept_no='d001') (cost=0.45 rows=2)
```



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

## EXPLAIN FORMAT: JSON Example

```
mysql> EXPLAIN FORMAT=JSON SELECT emp_no
-> FROM dept_manager WHERE dept_no='d001'\G
***** 1. row *****
EXPLAIN: {
 "query_block": {
 "select_id": 1,
 "cost_info": {
 "query_cost": "0.45"
 },
 "table": {
 "table_name": "dept_manager",
 "access_type": "ref",
 "possible_keys": [
 "PRIMARY",
 "emp_no",
 "dept_no"
],
 "key": "dept_no",
 "used_key_parts": [
 "dept_no"
],
 "key_length": "16",
 "ref": [
 "const"
],
 "rows_examined_per_scan": 2,
 "rows_produced_per_join": 2,
 "filtered": "100.00",
 "using_index": true,
 "cost_info": {
 "read_cost": "0.25",
 "eval_cost": "0.20",
 "prefix_cost": "0.45",
 "data_read_per_join": "64"
 },
 "used_columns": [
 "dept_no",
 "emp_no"
],
 "attached_condition": "(`employees`.`dept_manager`.`dept_no` = 'd001')"
 }
 }
}
```



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

## EXPLAIN ANALYZE

- It is available as of MySQL 8.0.18.
- It enables you to generate a plan for the query, instrument the query, and execute it without returning the query result.
- Output is similar to EXPLAIN FORMAT=TREE with additional actual statistics:
  - Time to return first row (in milliseconds)
  - Time to return all rows (in milliseconds)
  - Number of rows returned by the iterator
  - Number of loops

```
mysql> EXPLAIN ANALYZE SELECT emp_no FROM dept_manager WHERE dept_no='d001'\G
***** 1. row *****
EXPLAIN: -> Filter: (dept_manager.dept_no = 'd001') (cost=0.45 rows=2) (actual
time=0.020..0.025 rows=2 loops=1)
-> Index lookup on dept_manager using dept_no (dept_no='d001') (cost=0.45 rows=2)
(actual time=0.017..0.020 rows=2 loops=1)
```



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

## Hash Join Optimization

- Is available as of MySQL 8.0.18
- Applies to any query for which each join has an equi-join condition and uses no indexes
- Is faster than block-nested loop algorithm in such cases
- Builds a hash table on the first table and scans each rows in the second table to find the matching rows using the hash table
- Can be shown only by using EXPLAIN FORMAT=TREE and EXPLAIN ANALYZE

```
mysql> EXPLAIN FORMAT=TREE SELECT *
 -> FROM country JOIN countrylanguage ON UPPER(Code)=UPPER(CountryCode) \G
***** 1. row *****
EXPLAIN: -> Inner hash join (upper(country.`Code`) =
upper(countrylanguage.CountryCode)) (cost=23544.86 rows=235176)
 -> Table scan on countrylanguage (cost=0.42 rows=984)
 -> Hash
 -> Table scan on country (cost=25.40 rows=239)
```



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

## Topics

- Identifying slow queries
- EXPLAIN statement
- Working with indexes
- Index statistics
- MySQL Query Analyzer



ORACLE®

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

## Index Types

- **Nonunique:** Values can appear multiple times in the index.
  - This is the default index type.
- **Unique:** Values must be unique (or `NULL`).
- **Primary key:** Values are unique and cannot contain `NULL`.
  - Every table has at most one primary key.
- **Fulltext:** Values are character string data, and the index supports full text searching.
- **Spatial:** Values are spatial data types such as `GEOMETRY`, `POINT`, or `POLYGON`.
- **Functional:** Values are a result of expression or function based on table values.



ORACLE®

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

The index types shown in the slide are supported by InnoDB. Other storage engines might not support these index types.

Functions used in full text searching and spatial data handling are covered in the course titled *MySQL for Developers*.

## Creating Indexes to Improve Query Performance

- Enables efficient access to data rows
  - A query that searches for a value in an indexed field can immediately find rows containing that value.
    - A seek, which is efficient
  - A query that searches for a value in an unindexed field must read all rows to find rows containing that value.
    - A scan, which is inefficient
- Supports the following operations:
  - Direct value matching
    - Example: Find words such as “xenolith.”
  - Existence checking
    - Example: Determine that the word “xzzq” does not exist.
  - Range scans
    - Example: Find all words that begin with “xy.”



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

Consider the following analogy: If you are staying in a large hotel and you ask the receptionist if you have any printed messages, he or she looks in the specific mailbox assigned to your room to find them.

- If there is a message for you, the receptionist finds it in that specific location (a *seek*).
- If there is no message for you, that mailbox is empty. The receptionist does not need to scan all mailboxes for messages that are addressed to you.

Similarly, if you want to find all words in a dictionary that contain the letter *x*, you would need to go through all of the definitions (a *scan*). Alternatively, if you want to find all words beginning with the letter *x*, that is a much more efficient operation because all such words appear together in the dictionary. The dictionary is indexed in word order, with each word acting as the index key.

## Creating Indexes on Existing Tables

- To create a primary key when the table has no primary key:

```
ALTER TABLE tablename ADD PRIMARY KEY (col1, col2);
```

- To replace an existing primary key:

```
ALTER TABLE tablename DROP PRIMARY KEY, ADD PRIMARY KEY (col1, col2);
```

- To add a unique key:

```
ALTER TABLE tablename ADD UNIQUE (col3);
CREATE UNIQUE INDEX index2 ON tablename(col4);
```

- To add an ordered (nonunique) index:

```
ALTER TABLE tablename ADD INDEX (col5);
CREATE INDEX index3 ON tablename(col6);
```

- To add a functional index:

```
ALTER TABLE tablename ADD INDEX ((func(col7)));
CREATE INDEX index4 ON tablename((func(col8)));
```



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

CREATE INDEX and DROP INDEX are internally mapped to ALTER TABLE statements. You cannot use CREATE INDEX to create a primary key.

To create a functional index, the expression has to be enclosed with an additional pair of parentheses.

## Dropping Indexes on Existing Tables

- To drop a primary key:

```
ALTER TABLE table DROP PRIMARY KEY;
```

- To drop other types of indexes:

```
ALTER TABLE table DROP INDEX indexname;
DROP INDEX indexname ON table;
```

- Specify the index name to drop any unique, nonunique, fulltext, spatial, or functional indexes.
- To determine the index name, use SHOW INDEXES FROM *tablename* or SHOW CREATE TABLE *tablename*.



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

## Displaying Indexes Metadata

- SHOW CREATE TABLE *tablename* statement:

```
mysql> SHOW CREATE TABLE departments\G
***** 1. row *****
Table: departments
Create Table: CREATE TABLE `departments` (
 `dept_no` char(4) NOT NULL,
 `dept_name` varchar(40) NOT NULL,
 PRIMARY KEY (`dept_no`),
 UNIQUE KEY `dept_name` (`dept_name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

- The result shows all the indexes in the table; including the index name, index type, and the indexed columns.

- SHOW INDEXES FROM *tablename* statement:

- It returns the table index information, including some index statistics.
- Similar information can be obtained by querying the INFORMATION\_SCHEMA statistics view or executing the mysqlshow client program with the -k option.

ORACLE

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

### Example of SHOW INDEXES output:

```
mysql> SHOW INDEXES FROM departments;
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
Table	Non_unique	Key_name	Seq_in_index	Column_name	
Collation	Cardinality	Sub_part	Packed	Null	Index_type
Comment	Index_comment	Visible	Expression		
+-----+-----+-----+-----+-----+					
-----+-----+-----+-----+-----+					
-----+-----+-----+-----+-----+					
departments	0	PRIMARY	1	dept_no	
A	9	NULL	NULL	BTREE	
	YES	NULL			
departments	0	dept_name	1	dept_name	
A	9	NULL	NULL	BTREE	
	YES	NULL			
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

The Cardinality column provides an estimate of the number of unique values in the index.

## Invisible Indexes

- Enable you to “hide” indexes from the optimizer
- Test the effect of removing indexes on query performance
  - Without making destructive changes to drop the index
  - Avoids expensive operation to re-create the index if it is found to be required
- Continue to be updated by the server when data is modified by DML statements
- Cannot be applied to primary keys
- Should be marked as `INVISIBLE` in `CREATE TABLE`, `ALTER TABLE`, or `CREATE INDEX` statements
  - Can be “unhidden” by using the `VISIBLE` keyword



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

Invisible indexes make it possible to test the effect of removing an index on query performance, without making a destructive change that must be undone should the index turn out to be required. By marking an index as invisible, you are effectively "hiding" it from the optimizer while the index itself remains intact and can be restored at any time. This feature makes it much easier to test the removal of indexes and to perform a staged rollout of the changes.

Note that marking an index as invisible is not the same as disabling it. The index is still kept up-to-date and maintainable by DML statements.

Dropping and readding an index can be expensive for a large table, whereas making it invisible and visible are fast, in-place operations. Every index is visible by default. To mark it as invisible, you use the `INVISIBLE` keyword in a `CREATE TABLE`, `ALTER TABLE`, or `CREATE INDEX` statement. To restore the index visibility, execute another such statement with the `VISIBLE` keyword instead.

## Topics

- Identifying slow queries
- EXPLAIN statement
- Working with indexes
- **Index statistics**
- MySQL Query Analyzer



ORACLE®

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

## Maintaining InnoDB Index Statistics

- MySQL's optimizer uses index key distribution statistics to decide:
  - Which indexes MySQL uses for a specific table within a query
  - The join order when you perform a join on something other than a constant
- Update index statistics:
  - Automatically after 10% of rows change
  - Manually with `ANALYZE TABLE`



ORACLE®

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

## Automatically Updating Index Statistics

- InnoDB stores statistics persistently.
  - Enabled by default with the `innodb_stats_persistent` variable
  - Enabled per table with the `STATS_PERSISTENT` table option
  - Stored in the `mysql.innodb_index_stats` table
- Statistics update automatically.
  - Sampling 20 data pages by default
    - Changed with the `innodb_stats_persistent_sample_pages` variable
  - After 10% of rows change, enabled by default with the `innodb_stats_auto_recalc` variable
  - Optionally, after executing metadata statements such as `SHOW TABLE STATUS`, or when querying `INFORMATION_SCHEMA.TABLES`, enabled with the `innodb_stats_on_metadata` variable



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

If you disable persistent statistics gathering for one or more tables, InnoDB stores statistics for those tables in a volatile form internally, and uses the value of the `innodb_stats_transient_sample_pages` variable to decide how many pages it reads.

This is an example of the index statistics of the `salaries` table:

```
mysql> select index_name,stat_value,sample_size,stat_description from
 innodb_index_stats where database_name='employees' and table_name='salaries';
+-----+-----+-----+-----+
| index_name | stat_value | sample_size | stat_description |
+-----+-----+-----+-----+
PRIMARY	273180	20	emp_no
PRIMARY	2625007	20	emp_no,from_date
PRIMARY	5621	NULL	Number of leaf pages in the index
PRIMARY	5672	NULL	Number of pages in the index
emp_no	278300	20	emp_no
emp_no	2629176	20	emp_no,from_date
emp_no	2024	NULL	Number of leaf pages in the index
emp_no	2084	NULL	Number of pages in the index
+-----+-----+-----+-----+
```

## Using ANALYZE TABLE

- Analyzes and stores the key distribution statistics of a table
- Is used to make better choices about query execution
- Works with InnoDB, NDB, and MyISAM tables
- Supports partitioned tables
- ANALYZE TABLE option:
  - NO\_WRITE\_TO\_BINLOG or LOCAL: Suppress binary log
- Example of an ANALYZE TABLE result:

```
mysql> ANALYZE LOCAL TABLE salaries;
+-----+-----+-----+-----+
| Table | Op | Msg_type | Msg_text |
+-----+-----+-----+-----+
| employees.salaries | analyze | status | OK |
+-----+-----+-----+-----+
1 row in set (#.## sec)
```



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

ANALYZE TABLE locks the table with a read lock for the duration of the analysis.

## Rebuilding Indexes

- Use **FORCE**:

```
ALTER TABLE tablename FORCE
```

- Forces a rebuild even though the statement does not change the table structure
- Reorganizes data more evenly across data pages
- Reclaims file system space used by empty pages

- Rebuild full text indexes by using **OPTIMIZE TABLE**:

```
SET innodb_optimize_fulltext_only = 1;
OPTIMIZE TABLE tablename;
```

- By default, **OPTIMIZE TABLE** rebuilds the entire table.
- The **innodb\_optimize\_fulltext\_only** option ensures that **OPTIMIZE TABLE** rebuilds only the full text index.



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

Reorganize MyISAM and ARCHIVE tables by using the **OPTIMIZE TABLE** statement. On InnoDB, this statement executes by mapping to the **ALTER TABLE** statement shown in the slide.

## mysqlcheck Client Program

- Can be more convenient than issuing SQL statements
- Works with InnoDB, MyISAM, and ARCHIVE tables
- Three levels of checking:
  - Table specific
  - Database specific
  - All databases
- Some mysqlcheck maintenance options:
  - **--analyze**: Performs ANALYZE TABLE
  - **--check**: Performs CHECK TABLE (default)
  - **--optimize**: Performs OPTIMIZE TABLE
- Examples:

```
mysqlcheck -uroot -p employees employees salaries
mysqlcheck --login-path=admin --analyze --all-databases
```



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

## Histograms

- Provide the optimizer with extra information about data distribution within nonindexed columns
  - The optimizer knows how data is distributed for index columns, but not columns that do not form part of an index.
  - Skewed data might affect query performance.
- Approximate the data distribution within those columns
- Enable the optimizer to make better decisions



ORACLE®

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

To produce an optimal execution plan for a query, the optimizer benefits from knowing how the data is distributed in each column. It has some idea of this for columns that form part of an index, but it is clueless about the distribution of data in nonindexed columns. If a column contains skewed data, this can affect query execution.

You can create histograms on these columns to provide the optimizer an approximation of the data distribution within the columns. These help the optimizer to make more informed decisions about how to access the data they contain.

## Example: The Query

```
mysql> EXPLAIN SELECT * FROM orders
 JOIN customer ON o_custkey = c_custkey
 WHERE o_orderdate < '1993-01-01' AND c_mktsegment = "AUTOMOBILE"\G

 1. row *****
 id: 1
 select_type: SIMPLE
 table: customer
 partitions: NULL
 type: ALL
 possible_keys: PRIMARY
 key: NULLz
 key_len: NULL
 ref: NULL
 rows: 149050
 filtered: 10.00
 Extra: Using where

 2. row *****
...
2 rows in set, 1 warning (#.## sec)
```

```
mysql> SELECT count(*) FROM orders
 JOIN customer ON o_custkey = c_custkey
 WHERE o_orderdate < '1993-01-01' AND c_mktsegment = "AUTOMOBILE"\G

 1. row *****
count(*): 45127
1 row in set (49.98 sec)
```

ORACLE®

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

To understand how histograms affect query execution, consider this example. This is a query that joins the orders and customer tables for orders of automobiles in 1993 and later. Note from the EXPLAIN output in column one that you must perform a full table scan on the CUSTOMER table and the selectivity (filtered column in the EXPLAIN output) is 10%. The selectivity of a column is the “uniqueness” of its values. High selectivity implies high uniqueness, or a low number of matching values. Low selectivity implies a low uniqueness, or a high percentage of matches. In this example, many rows in the CUSTOMER table have "AUTOMOBILE" in the c\_mktsegment column. When you execute a count of the number of rows involved in the query, it takes approximately 50 seconds.

## Example: Creating a Histogram

```
mysql> ANALYZE TABLE customer
-> UPDATE HISTOGRAM ON c_mktsegment WITH 1024 BUCKETS;
+-----+-----+-----+-----+
| Table | Op | Msg_type | Msg_text |
+-----+-----+-----+-----+
| dbt3.customer | histogram | status | Histogram statistics created |
| | | | for column 'c_mktsegment'. |
+-----+-----+-----+-----+
1 row in set (#.## sec)
```



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

To provide the optimizer with more insight into the data distribution of the `c_mktsegment` column in the `CUSTOMER` table, create a histogram. You use the `ANALYZE TABLE ... UPDATE HISTOGRAM` syntax to do this, specifying a number of “buckets” for the data to be sorted into. A histogram sorts values into “buckets,” as you might sort coins into buckets. You can have up to 1,024 such buckets but creating too many is often counter-productive. So, typically, you would start with a low number first, and then test and refine.

## Example: The Query with Histogram Data

```
mysql> EXPLAIN SELECT * FROM orders
 JOIN customer ON o_custkey = c_custkey
 WHERE o_orderdate < '1993-01-01' AND c_mktsegment = "AUTOMOBILE"\G

1. row ****
...

2. row ****
 id: 1
 select_type: SIMPLE
 table: customer
 partitions: NULL
 type: eq_ref
possible_keys: PRIMARY
 key: PRIMARY
 key_len: 4
 ref: dbt3.orders.o_custkey
 rows: 1
filtered: 19.84
 Extra: Using where
2 rows in set, 1 warning (#.## sec)
```

```
mysql> SELECT count(*) FROM orders
 JOIN customer ON o_custkey = c_custkey
 WHERE o_orderdate < '1993-01-01' AND c_mktsegment = "AUTOMOBILE"\G

1. row ****
count(*): 45127
1 row in set (6.35 sec)
```



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

After creating the histogram, you can see that the optimizer chooses not to start with the CUSTOMER table because almost twice as many rows (19.84%) will cause look-ups into the ORDERS table.

This query now executes much faster: under 6.5 seconds compared to 50 seconds before creating the histogram.

## Topics

- Identifying slow queries
- EXPLAIN statement
- Working with indexes
- Index statistics
- MySQL Query Analyzer



ORACLE®

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

## MySQL Query Analyzer

- Monitors SQL statements executed on the MySQL server:
  - Nature of query
  - Number of executions
  - Execution times
- Extracts this information from the Performance Schema by using the MySQL Enterprise Monitor Agent by default
  - Can configure client applications to route database requests via MySQL Proxy and MySQL Aggregator
  - Can install a Connector plugin to send this information directly to the MySQL Enterprise Monitor Service Manager
- Normalizes queries for easier reporting
  - Combines similar queries with different literal values
- Enables you to drill into each query for detailed information



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

**Note:** MySQL Query Analyzer does not track server-side prepared statements. However the default configurations for most client-side libraries for MySQL don't use them. They emulate them on the client-side, and those will be tracked by the query analyzer.

## Query Response Time Index

MySQL Query Analyzer generates a Query Response Time Index (QRTi) for each query.

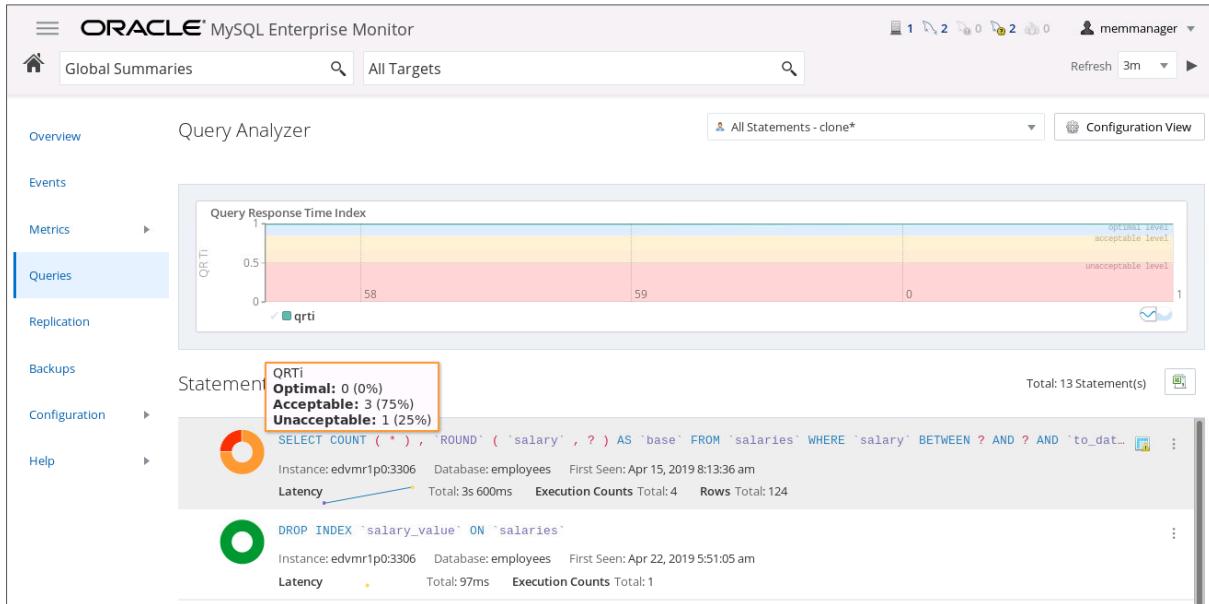
- Enables an “at-a-glance” indicator of query performance
- Is generated by an algorithm that considers the number of executions and duration of each query:

| Status              | Default time value             | Assigned Value | Meaning                                                                           |
|---------------------|--------------------------------|----------------|-----------------------------------------------------------------------------------|
| <b>Optimum</b>      | 100ms                          | 1.0 (100%)     | All instances of the query were within the acceptable response time.              |
| <b>Acceptable</b>   | 4 * Optimum:<br>100ms to 400ms | 0.5 (50%)      | Query execution time was less than 400 ms (the default acceptable response time). |
| <b>Unacceptable</b> | > Acceptable                   | 0 (0%)         | All instances of the query exceeded the acceptable response time.                 |



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

## Query Analyzer User Interface



ORACLE®

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

## Quiz



Which queries are most important to optimize?

- a. Complex queries
- b. Queries that cannot use a WHERE clause
- c. Queries that execute most frequently
- d. Queries that take the most time to execute



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

**Answer: c**

## Summary



In this lesson, you should have learned how to:

- Identify queries suitable for optimization
- Examine query index usage
- Create indexes to improve server performance
- Maintain index statistics
- Use MySQL Query Analyzer



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

## Practices

- 9-1: Improving Query Performance with Indexes
- 9-2: Using MySQL Query Analyzer



ORACLE®

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

10

# Choosing a Backup Strategy



ORACLE®

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

## Objectives



After completing this lesson, you should be able to:

- Distinguish between the different types of backup
- State the advantages and disadvantages of the various backup techniques
- Implement a backup strategy



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

## Topics

- Understanding backups
- Backup techniques
- Creating a backup strategy



ORACLE®

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

## Reasons to Back Up

- Database recovery
  - Restoring from complete system failure
  - Recovering certain portions of the system to a state prior to a user error
- Auditing and analysis
  - Creating an environment separate from the primary production environment for data audit or analysis
- Typical DBA tasks
  - Transferring data from one system to another
  - Creating a development server based on a specific state of a production server



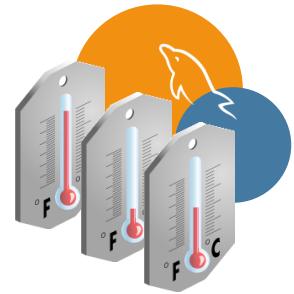
ORACLE®

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

## Backup Types

Backup types affect how applications interact with data during the backup operation:

- **Hot** backups generally allow applications full access to the data.
- **Cold** backups generally does not allow applications to access data.
- **Warm** backups permit applications to read, but not to modify data.



ORACLE®

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

## Hot Backups

- Hot backups take place while the data is being read and modified with little to no interruption of your ability to interface with or manipulate data.
- The system remains accessible for operations that read and modify data.
- Lock data by:
  - Using multiversion concurrency control (MVCC)
  - Locking at a low level (such as row level)
  - Not locking at all so that applications can continue accessing the data



ORACLE®

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

Multiversion concurrency control (MVCC) is a feature of InnoDB that uses the undo log to maintain a consistent snapshot of what the data looked like at a point in time.

## Cold Backups

- Take place while the data is inaccessible to users
  - Typically the server is in an inaccessible mode or shut down entirely during a cold backup.
  - You cannot read or make any modifications to the data while the backup takes place.
- Prevent you from performing any activities with the data
- Do not interfere with the performance of the system when it is up and running
  - However, having to lock out or completely block user access to data for an extended period of time is not acceptable for some applications.



ORACLE®

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

## Warm Backups

- Take place while the data is being accessed
  - In most cases, the data cannot be modified while the backup is taking place.
- Have the advantage of not having to completely lock out end users
  - However, the disadvantage of not being able to modify the data while the backup is taking place can make this type of backup not suitable for certain applications.
- Might result in performance issues because you cannot modify data during the backup.
  - Updating users may be blocked for a long duration of time.



ORACLE®

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

## Quiz



Which backup type takes place while the data is being read and modified with little to no interruption to your ability to interface or manipulate data?

- a. Hot backup
- b. Cold backup
- c. Warm backup



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

**Answer: a**

## Topics

- Understanding backups
- **Backup techniques**
- Creating a backup strategy



ORACLE®

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

## Backup Techniques

- Logical backup:
  - Textual representation: SQL statements or data files in the form of comma-separated files, tab delimited files, or XML files
  - Results in a text file containing SQL statements or data to reconstruct the database
- Physical backup:
  - Binary copy of the MySQL database files
- Snapshot based (physical backup)
- Replication based (either logical or physical backup)
- Incremental backup:
  - Created by copying and flushing the MySQL binary logs



ORACLE®

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

## Logical Backups

Perform a complete data dump by using SQL statements, `mysqldump` or `mysqlpump`.

- These data dumps are based on a specific point in time but are the slowest of all the backup techniques.
- Advantage:
  - The process creates a SQL script that you can execute on a MySQL server or data files that you can import into database tables.
  - You can use the script to reload the database on another host, running a different architecture or different version of MySQL Server.
- Disadvantage:
  - By default (and always for non-InnoDB tables), `mysqldump` and `mysqlpump` lock tables during the dump, which prevents users from modifying data during the backup.



ORACLE®

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

## Logical Backups

- Logical backups:
  - Convert databases and tables to text files containing SQL statements or text data
  - Can back up all databases, one or more databases, or one or more tables
  - Are portable
  - Require that the MySQL server is running during the backup
  - Can back up both local and remote MySQL servers
  - Are generally slower than physical (binary) backups
- The size of a logical backup file might exceed the size of the database being backed up.
  - The statements might take up more space than the data they represent.
  - Textual data usually take up more space than binary data.
  - However, because InnoDB stores data in data pages, which can contain unused space, the InnoDB data files often take considerably more space on disk than the data requires.



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

## Logical Backup Conditions

- Logical backups are usually warm:
  - The MySQL server must be running when you create logical backups.
  - Other applications can perform read operations during the logical backup.
  - The server creates the files by reading the structure and contents of the tables being backed up.
  - It then converts the structure and data to SQL statements or text files.
- Hot logical backups are possible with InnoDB MVCC:
  - Start a transaction in the repeatable read isolation level to back up a consistent point-in-time copy of InnoDB tables.
- Using logical backups, you can back up local and remote MySQL servers.
  - Physical backups can be performed only on the server host.



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

## Logical Backup Performance

- Logical backups are generally slower than physical backups.
  - The MySQL server must read tables and interpret their contents.
  - It must then translate the table contents to a disk file or send the statements to a client program, which writes them out.
  - The `mysqlpump` client utility performs better than `mysqldump` because it processes databases and their objects in parallel, but it is still slower than a physical backup.
- Recovering from a logical backup is slower than recovering from a physical backup.
  - The recovery process executes the script containing the individual `CREATE` and `INSERT` statements that create each backed-up table and row.
  - All the indexes in the recovered table must be rebuilt.



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

## Physical Backups

- Are created by copying the data files
  - Use standard commands such as `tar`, `cp`, `cpio`, `rsync`, or `xcopy`.
  - Use physical mirroring or block device online disk copying.
  - Use snapshot-based file archives.
- Must be restored to the same storage engine and MySQL version
  - When you recover a physical MySQL backup from an InnoDB table, it remains an InnoDB table on the target server.
- Can be restored across machine architectures
  - The files must be binary portable on the storage engine level.
- Are faster to perform and restore than logical backups and recoveries



ORACLE®

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

## Physical Backup Files

- Physical binary backups are faster than logical backups because the process is a simple file or filesystem copy.
- A physical backup is an exact representation of the bits in the database files.
  - These copies preserve the databases in exactly the same format in which MySQL itself stores them on disk.
  - Because they are exact copies of the originals, physical backups are the same size as the original files.



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

## Physical Backup Conditions

- The server must not modify data files during backup.
  - If you copy the live data files, prevent writes to those files:
    - For InnoDB: MySQL server shutdown is required.
    - For MyISAM: Lock tables to allow reads but not changes.
- You can also minimize the effect to MySQL and applications by using techniques that separate the data files being backed up from the MySQL server:
  - Snapshots
  - Replication
  - DRBD or other methods that copy the whole filesystem



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

## Online Disk Copies

- You can back up data directly to another disk by using:
  - Processes such as storage replication or RAID mirroring
  - External applications such as DRBD
    - Distributed Replicated Block Device
    - Enables low-level disk copying between clustered computers
- These techniques provide
  - Live (or nearly live) backups
  - A quick means of recovering data in the event of a hardware failure
- Disadvantage:
  - Replicas are created in real time (or near real time), so you cannot use this technique to recover from data loss caused by user or application errors.



ORACLE®

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

## Snapshot-Based Backups

- Create a point-in-time copy of the data
  - You typically perform a physical backup against the snapshot copy.
- Provide a logically frozen version of the filesystem from which you can take MySQL backups
  - The frozen filesystem does not necessarily contain a consistent database image.
  - When you recover from such a filesystem, InnoDB must perform its own recovery to ensure there are no incomplete transactions.
- May require MySQL Server to be shut down (for InnoDB tables) or tables to be locked (for MyISAM tables)
  - To ensure data consistency
  - Data files are stored in multiple volumes where multiple snapshots are required.
- Greatly reduce the time during which the database and applications are unavailable



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

## Performing a Snapshot

- Snapshot-based backups use a snapshot capability that is external to MySQL.
  - Example: If your data files are on an LVM2 logical volume on Linux that contains an appropriate filesystem such as ext4, you can create snapshot backups.
- Snapshot-based backups work best for transactional engines that perform their own recovery, such as InnoDB.
- The time needed to perform the snapshot does not increase as the size of the database grows.
  - The backup window (from the perspective of the application) is almost zero.
  - Snapshots use a copy-on-write method to ensure they are almost instantaneous.
    - Cause additional I/O overhead
  - There is a performance cost while the snapshot is active.



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

“Copy-on-write” is a technique that copies data blocks on disk when they change and is similar to the method used by the InnoDB undo log and MVCC. When you perform a snapshot, LVM marks all blocks that subsequently change and records the snapshotted version of those blocks. The initial snapshot takes a little time, but all subsequent write operations incur additional performance and storage cost while the snapshot is active.

## Replication-Based Backups

- MySQL supports one-way asynchronous replication, in which one server acts as the master while one or more other servers act as slaves.
- MySQL replication can be used for backups:
  - The master is used for the production applications.
  - A slave is used for backup purposes.
- This eliminates the impact on production applications.
- The backup of the slave is either logical or physical.
- Higher cost: There must be another server and storage to store the replica of the database.
- Based on asynchronous replication:
  - The slave may be delayed with respect to the master.
  - This is acceptable if the binary log is not purged before the slave has read it.



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

## Binary Log Backups

Binary log backups record modifications to the data.

- Useful for minimizing exposure between full backups by restoring events that have occurred since the last full backup
- Advantages:
  - Binary log backups contain a record of all changes to the data over time.
    - Other backup types contain a snapshot of the data.
  - You can take multiple binary log backups in sequential order.
- Disadvantages:
  - You must sequentially restore all sequential binary logs that were created from the last full backup.
  - Recovery from a system failure can be slow depending on the number of binary logs that must be executed.



ORACLE®

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

`mysqlbinlog` can stream the binary logs from a MySQL server as live backups.

```
mysqlbinlog --read-from-remote-server --host=server_host --raw
--stop-never binlog.000130
```

See <https://dev.mysql.com/doc/refman/en/mysqlbinlog-backup.html> for `mysqlbinlog` options to backup binary logs.

## Binary Logging and Incremental Backups

- Control binary logging at the session level:

```
SET SQL_LOG_BIN = {0|1|ON|OFF}
```

- Binary log has been enabled on the MySQL server.
  - You must have the SUPER privilege to set this variable.

- Flush binary logs when taking logical or physical backups.
  - Synchronize binary logs to backups.
- Logical and physical backups are full backups.
  - All rows of all tables are backed up.
- To perform an incremental backup, copy binary logs.
- Binary logs can be used for point-in-time recovery.
  - You can identify transactions that caused damage and skip them during restore.



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

## Quiz



Which backup type converts databases and tables to SQL statements?

- a. Logical backup
- b. Physical backup
- c. Replication-based backup
- d. Snapshot-based backup



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

**Answer: a**

## Topics

- Understanding backups
- Backup techniques
- Creating a backup strategy



ORACLE®

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

## Comparing Backup Methods

| Method                                              | Hot/Warm/<br>Cold             | Storage<br>Engines | Logical/<br>Physical   | Consistent | Availability                          |
|-----------------------------------------------------|-------------------------------|--------------------|------------------------|------------|---------------------------------------|
| MySQL Enterprise<br>Backup                          | Hot (InnoDB)/<br>warm (other) | All                | Physical               | Yes        | Commercially<br>available             |
| <code>mysqldump</code> or<br><code>mysqlpump</code> | Hot (InnoDB)/<br>warm (other) | All                | Logical                | Yes        | Freely available                      |
| Snapshots                                           | Hot                           | All                | Physical               | Yes        | Need snapshot<br>volume or filesystem |
| Replication                                         | Hot                           | All                | Logical or<br>physical | Yes        | Freely available                      |
| SQL Statements                                      | Warm                          | All                | Logical                | No         | Freely available                      |
| OS Copy<br>Commands                                 | Cold or warm                  | All                | Physical               | Yes        | Freely available                      |



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

### **`mysqldump` and `mysqlpump`**

These utilities can perform a hot backup on InnoDB tables only when the entire operation is wrapped in a transaction by specifying the `--single-transaction` option.

### **Snapshots**

Snapshots do not work in the same way for all engines. For example, InnoDB tables do not require `FLUSH TABLES WITH READ LOCK` to start a snapshot, but MyISAM tables do.

## Deciding a Backup Strategy

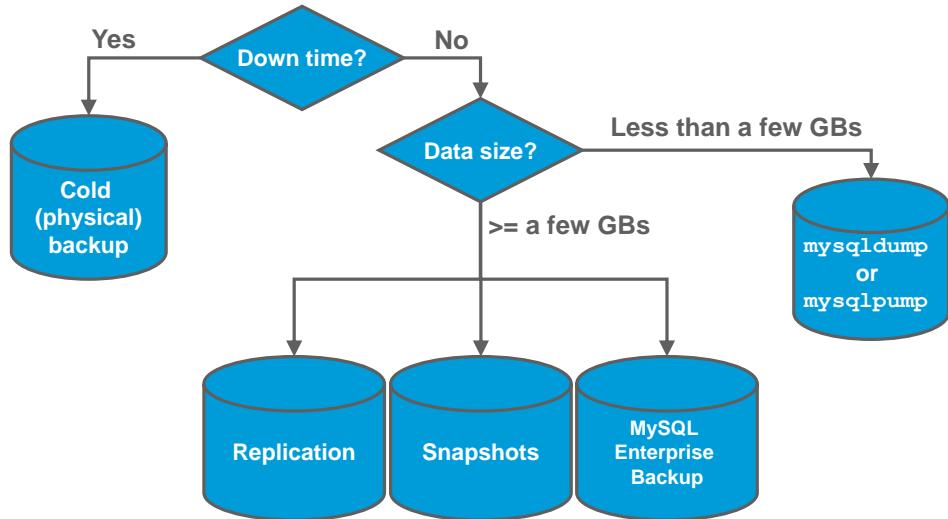
- The backup and recovery strategy that you implement is determined by:
  - How complete is the data record?
    - You might choose not to back up every table or all data in each table.
  - How current is the backup?
    - Some data does not change very often, so you might choose to back up some tables (or portions of some tables) infrequently.
- Other questions to ask when deciding your strategy:
  - How much down time can your system afford, if any?
  - How much data is there to back up?
  - Which storage engines are being used to store the data (InnoDB, MyISAM, or both)?



ORACLE®

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

## Backup Strategy: Decision Chart



ORACLE®

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

## More Complex Strategies

Combine multiple backup techniques to create more complex strategies:

- Example using full and binary log backups:
  - Nightly backups using `mysqlbackup` from a replication slave
  - Multiple binary log backups each hour to minimize exposure during each day
- Example using partial backups:
  - Technique:
    - `mysqldump` with `--where` option
    - `SELECT INTO OUTFILE`
  - Weekly conditional backups of large transactional tables containing only fixed or historical data
    - Data that does not change, for example, fulfilled orders that have passed the “return by” date
  - Daily/hourly supplemental backups containing live data



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

If you can be sure that historical data before a threshold date does not change, you do not need to back it up more than once. Backing up only data after that point is faster, although it means you must take care that your backups represent a consistent view of the data, and the restore operation is more complex.

## Summary



In this lesson, you should have learned how to:

- Distinguish between the different types of backups
- State the advantages and disadvantages of the various backup techniques
- Implement a backup strategy



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

## Practices

- 10-1: Quiz—Backup Strategies



ORACLE®

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

11

# Performing Backups



ORACLE®

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

## Objectives



After completing this lesson, you should be able to:

- Use MySQL Enterprise Backup to perform consistent backups
- Use the `mysqldump` and `mysqlpump` utilities to perform logical backups
- Explain when and how to use physical file backups
- Back up from a replication slave
- Use binary logs to recover the database



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

## Topics

- MySQL backup tools
  - Physical backups using MySQL Enterprise Backup
  - Logical backups using `mysqldump` and `mysqlpump`
  - Other physical backup methods
  - Replication-based backups
  - Recover database using binary logs



ORACLE®

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

## Backup Tools: Overview

- SQL statements for logical backups
- SQL statements combined with operating system commands for physical backups
  - Example: `LOCK TABLES`
- Other backup tools for MySQL:
  - MySQL Enterprise Backup : Physical backups
  - `mysqldump`: Logical backups
  - `mysqlpump`: Logical backups
- Third-party tools



ORACLE®

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

Oracle commercial and community tools are the primary focus of this lesson. There are commercial and community third-party tools that you can incorporate into your backup strategies.

## Topics

- MySQL backup tools
- Physical backups using MySQL Enterprise Backup
- Logical backups using mysqldump and mysqlpump
- Other physical backup methods
- Replication-based backups
- Recover database using binary logs



ORACLE®

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

## MySQL Enterprise Backup

- Multi-platform hot backup tool for MySQL
  - Run from the command line: `mysqlbackup`
- Optimized for InnoDB tables
  - Capable of backing up tables of other storage engines supported by MySQL
- Supports:
  - Incremental and differential backups in addition to the normal full backup
  - Single-file backups:
    - Can be streamed to on another storage device or file server
    - Back up to tape
    - Direct cloud storage backup
  - Backup encryption and compression
  - Partial backups and transportable tablespace
- Backups can be monitored in MySQL Enterprise Monitor backup dashboard.



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

For MySQL Server 8.0, use the MySQL Enterprise Backup version with the same version number as the server.

## MySQL Enterprise Backup: Storage Engines

- InnoDB
  - MySQL Enterprise Backup performs **hot** backups on InnoDB tables.
    - The backup takes place while applications connected to the database are running.
    - This type of backup does not block normal database operations.
      - It captures even changes that occur while the backup is happening.
- Other storage engines
  - MySQL Enterprise Backup performs a **warm** backup.
  - The database tables can be read by applications.
  - While the non-InnoDB backup runs:
    - Prior to version 8.0.16, the database cannot be modified
    - As of version 8.0.16, only non-InnoDB tables cannot be modified and InnoDB tables allow DML operations but not DDL operations



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

MySQL Enterprise Backup does not support offline (cold) backups. The MySQL server must be running for `mysqlbackup` to connect to it to start a backup.

## MySQL Enterprise Backup: InnoDB Files

Raw InnoDB files that are backed up with `mysqlbackup`:

- `ibdata*` files: Shared tablespace files that contain the system tablespace and possibly the data for some user tables
- `mysql.ibd`: mysql tablespace containing the data dictionary
- `.ibd` files: Per-table data files and general tablespace files
- `undo_*` files: Undo log tablespaces
- `ib_logfile*` files
  - Data extracted from the `ib_logfile*` files is stored in a new backup file named `ibbackup_logfile`.
  - This includes redo log information representing changes that occur while the backup is running.



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

## MySQL Enterprise Backup: Non-InnoDB Files

- MySQL Enterprise Backup is optimized for InnoDB.
- You can use MySQL Enterprise Backup to back up non-InnoDB data, if the following are true:
  - The MySQL server you want to back up supports InnoDB
  - The server contains at least one InnoDB table
- By default, `mysqlbackup` backs up all files in subdirectories under the data directory.
  - If you specify the `--only-known-file-types` option, the backup includes only additional files with MySQL-recognized extensions.
- For example, the following files are backed up for data stored with MyISAM:
  - `.MYD`: MyISAM data files
  - `.MYI`: MyISAM index files
  - `.sdi`: Metadata files



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

## Full Backups

- Start `mysqlbackup` on the MySQL server host, not remotely.

```
mysqlbackup --user=username --password --port=portnumber
--backup_dir=backup-directory command
```

- Provide credentials and server coordinates using the same options as the `mysql` client: including `--user`, `--port`, and `--password`
  - The `--backup_dir` option specifies the directory to store the backup data and metadata.
- Basic commands include:
    - `backup`: Performs the initial phase of a backup
    - `backup-and-apply-log`: Includes the initial phase of the backup and a second phase that brings the InnoDB tables in the backup up-to-date, including any changes made to the data while the backup was running



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

## Single-File Backups

- Use the `backup-to-image` command to write the backup into a single file:

```
mysqlbackup -uuser -ppassword --backup_dir=temp-backup-dir
--backup-image=image-file backup-to-image
```

- The `--backup-image` option specifies the file name and path to store the backup.
- The `--backup_dir` option specifies a temporary folder to store the backup metadata.

- Advantages of single file backups:

- Easier to manage
  - One file for each backup
- Can stream to another device or server
- Can back up to tape
- Can back up to cloud storage



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

In many UNIX commands, the dash symbol “-” represents standard output. When you use it in the option `--backup-image=-`, it sends the backup to standard output, from which you can redirect it with standard output redirection with “`> image-file`” as in this example:

```
mysqlbackup -uuser -ppassword --backup_dir=backup-dir
--backup-image=- backup-to-image > image-file
```

## Backup Process

1. mysqlbackup opens a connection to the MySQL server.
2. mysqlbackup takes an online backup of InnoDB tables.
  - This phase does not disturb normal database processing.
3. When the mysqlbackup run has almost completed, it:
  - Executes the SQL statement `LOCK INSTANCE FOR BACKUP`
    - This blocks DDL operations while allowing DML operations to continue.
  - Executes the SQL statement `FLUSH TABLES tbl_name,...`  
`WITH READ LOCK` on all non-InnoDB tables that are included in the backup
  - Copies the non-InnoDB files (such as MyISAM .MYI and .MYD files) to the backup directory
4. mysqlbackup runs to completion and UNLOCKS the tables and instance.



ORACLE®

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

The read-locked phase lasts only a short time if:

- You do not run long `SELECT` or other queries in the database while the read lock is in effect
- Your non-InnoDB tables are small

Step 3 in the slide describes the process for mysqlbackup 8.0.16 and newer.

For mysqlbackup prior to 8.0.16, Step 3 will lock the whole database with the SQL statement `FLUSH TABLES WITH READ LOCK`, which blocks any write operations on all tables.

## Incremental Backups

Each **incremental backup** only includes the changes since the previous backup, which can itself be a full or incremental backup.

```
mysqlbackup --user=username --password --port=portnumber
 --incremental --incremental-base=history:last_backup
 --backup_dir=incr-backup-directory backup
```

- Add the `--incremental` option to perform an incremental backup
- Use the `--incremental-base` option to indicate the previous backup as the base for the incremental backup
- Can increase the speed of backup and reduce the storage required to store backups if large portions of the data remain unchanged in between the time of two backups
- Applies to InnoDB tables or non-InnoDB tables that are read only or rarely updated
  - For non-InnoDB files, the entire file is included in an incremental backup if that file has changed since the previous backup.



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

You can back up to a single file by replacing the `backup` command with the `backup-to-image` command and specify the `--backup-image` option.

The `--incremental-base=history:last_backup` option causes `mysqlbackup` to query the `end_lsn` value from the last successful non-TTS backup as recorded in the `backup_history` table of the server instance that is being backed up.

Alternatively, you can specify the value `dir:directory_path` as the value for the `--incremental-base` option to specify a directory path containing a previous backup.

Instead of the `--incremental-base` option, you can use the `--start-lsn` option to specify the highest LSN value included in a previous backup to be used as the base of the incremental backup.

## Differential Backups

Each **differential backup** includes all the changes made to the data since the last full backup. This is a special case of incremental backup that has a full backup as its base.

```
mysqlbackup --user=username --password --port=portnumber
 --incremental --incremental-base=history:last_full_backup
 --backup_dir=incr-backup-directory backup
```

- Use the `--incremental-base=history:last_full_backup` option to specify the last full backup as the base of the incremental backup. This option is available as of 8.0.17.
- Similar to incremental backup except the base is always a full backup



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

You can back up to a single file by replacing the `backup` command with the `backup-to-image` command and specify the `--backup-image` option.

Instead of the `--incremental-base` option, you can use the `--start-lsn` option to specify the highest LSN value included in a previous full backup to be used as the base of the differential backup. If you specify a highest LSN value of a previous incremental backup instead of a full backup, the backup becomes an incremental backup instead of a differential backup.

**Note:** The commands for incremental and differential backups are similar; the base backup that you specified determines whether the backup is incremental or differential.

## Validate Operations

The `validate` command checks the integrity of the backups.

- Validating a backup image:

```
mysqlbackup --backup-image=image-file validate
```

- Validating a backup directory:

```
mysqlbackup --backup-dir=backup-directory validate
```

- Validates the checksum value for each data page in the backup.

- Limitations:

- It only validates the InnoDB data files (`ibdata*` and `*.ibd` files).
- It cannot detect missing pages that have been removed or truncated from any `*.ibd` files.
- It cannot detect missing files or files that have been deleted from the backup.



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

## Restore Operations

1. Shut down the MySQL server.
2. Delete all files inside the server's data directory.
  - Delete all files inside the directories specified by the `--innodb_data_home_dir`, `--innodb_log_group_home_dir`, and `--innodb_undo_directory` options.
3. Run `mysqlbackup` to restore the backup files from a full backup.
4. Optionally, restore from any incremental or differential backups if available.
5. Apply available binary logs to:
  - Recover all changes to the database after the last backup
  - Perform a point-in-time recovery to prevent wrong operations from corrupting the data
6. Start the MySQL server.



ORACLE®

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

## Restore Commands

- **copy-back**

- Restores a directory backup that has been made consistent using the apply-log command

```
mysqlbackup --backup-dir=backup-dir copy-back
```

- Restores the files from the *backup-dir* to the data directory

- **copy-back-and-apply-log:**

- In a single step, restores a single-file backup or a directory backup and performs an apply-log operation to the restored data

```
mysqlbackup --backup-dir=temp-backup-dir
--backup-image=image-file copy-back-and-apply-log
```

- Restores the files from the *image-file* to the data directory using the *temp-backup-dir* for storing temporary files



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

When you use the `copy-back` command, MySQL Enterprise Backup:

- Copies the data files, logs, and other backed-up files from the backup directory to their original locations
- Performs any required post-processing on them

During the `copy-back` process, `mysqlbackup` cannot query its settings from the server, so it reads the standard configuration files for options such as the `datadir`.

- If you want to restore to a different server, you can provide a non-standard defaults file with the `--defaults-file` option.

## Restoring Incremental Backups

- You must restore the correct base backup before restoring the incremental or differential backup.
- Add the `--incremental` option to the restore operation
  - From an incremental backup directory:

```
mysqlbackup --incremental-backup-dir=incr-backup-dir
 --incremental copy-back-and-apply-log
```

- From a single-file incremental backup:

```
mysqlbackup --backup-dir=temp-backup-dir --backup-image=image-file
 --incremental copy-back-and-apply-log
```



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

## Update Operations

A directory backup of a full backup can be updated using:

- The apply-log operation

```
mysqlbackup --backup-dir=backup-dir apply-log
```

- A full backup taken using the `backup` command that has been updated using the `apply-log` command can be restored directly using the `copy-back` command instead of the `copy-back-and-apply-log` command.

- The apply-incremental-backup operation

```
mysqlbackup --incremental-backup-dir=incr-backup-dir
--backup-dir=backup-dir apply-incremental-backup
```

- A full backup in the `backup-dir` can be updated with the incremental backup that uses the full backup as the base.
- This can speed up the restore process as it can reduce the number of incremental backups that need to be restored. This can also speed up backup as it can convert an incremental backup into a full backup.



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

## Single-File Operations

- Convert an existing backup directory to a single-file backup:

```
mysqlbackup --backup_dir=backup-dir
 --backup-image=image-file backup-dir-to-image
```

- List the contents of a single-file backup:

```
mysqlbackup --backup-image=image-file list-image
```

- Extract files from a single-file backup:

```
mysqlbackup --backup-image=image-file --src-entry=file-to-extract
 --dst-entry=file-to-extract extract
```

- **--src-entry**: Identifies a file or directory to extract from a single-file backup
- **--dst-entry**: Is used with single-file backups to extract a single file or directory to a user-specified path



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

## Basic Privileges Required for MySQL Enterprise Backup

- Minimum privileges required:
  - BACKUP\_ADMIN (for 8.0.16 and later) and RELOAD on all databases and tables
  - CREATE, INSERT, UPDATE, and DROP on mysql.backup\_progress table
  - CREATE, INSERT, UPDATE, DROP, SELECT, and ALTER on mysql.backup\_history table
  - SUPER to enable and disable logging and to optimize locking in order to minimize disruption to database processing
  - REPLICATION CLIENT to retrieve the binlog position stored with the backup
  - PROCESS to process DDL statements with ALGORITHM=INPLACE clause
  - SELECT on performance\_schema.replication\_group\_members, to know whether the server instance is part of a Group Replication setup
  - For 8.0.16 and later, SELECT on performance\_schema.variables\_info and performance\_schema.log\_status



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

Performing backup operations using MySQL Enterprise Backup requires certain privileges.

- Use an existing MySQL root account
- Or create a new account with sufficient privileges.

The mysqlbackup client connects to the server with the --user and --password options.

Specifying login details at the command line is not ideal, so consider including them in the [client] section of the mysqlbackup my.cnf file (or other configuration file). mysql\_config\_editor can be used to create an encrypted password in the .mylogin.cnf file.

Refer to <https://dev.mysql.com/doc/mysql-enterprise-backup/en/mysqlbackup.privileges.html> for other privileges required in some special scenarios when certain features are used.

## Granting Required Privileges

The following script sets the required permissions for the backupuser user:

```
CREATE USER 'mysqlbackup'@'localhost' IDENTIFIED BY 'password';
GRANT BACKUP_ADMIN, RELOAD ON *.* TO 'mysqlbackup'@'localhost';
GRANT CREATE, INSERT, DROP, UPDATE ON mysql.backup_progress TO
 mysqlbackup'@'localhost';
GRANT CREATE, INSERT, DROP, UPDATE, SELECT, ALTER ON mysql.backup_history TO
 'mysqlbackup'@'localhost';
GRANT SUPER, REPLICATION CLIENT, PROCESS ON *.* TO 'mysqlbackup'@'localhost';
GRANT SELECT ON performance_schema.replication_group_members TO
 'mysqlbackup'@'localhost';
GRANT SELECT ON performance_schema.variables_info TO 'mysqlbackup'@'localhost';
GRANT SELECT ON performance_schema.log_status TO 'mysqlbackup'@'localhost';
```



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

## Quiz



Which InnoDB files are backed up with the `mysqlbackup` command?

- a. `ibdata*` files
- b. `.ibd` files
- c. `undo_*` files
- d. `ib_logfile*` files
- e. All of the above



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

**Answer: e**

## Topics

- MySQL backup tools
- Physical backups using MySQL Enterprise Backup
- **Logical backups using mysqldump and mysqlpump**
- Other physical backup methods
- Replication-based backups
- Recover database using binary logs



ORACLE®

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

## **mysqldump and mysqlpump**

- The `mysqldump` and `mysqlpump` utilities are included in the MySQL distribution.
- They perform logical backups and work with any database engine.
  - Certain features such as “hot” backups can be performed only on InnoDB tables.
  - By default, they run as “warm” backups for all storage engines.
- They can be automated by using `crontab` in Linux and UNIX and the Windows Task Scheduler in Windows.
- There are no tracking or reporting tools for `mysqldump` or `mysqlpump`.



ORACLE®

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

## `mysqldump`

- Dumps table contents to files:
  - All databases, specific databases, or specific tables
  - Allows backup of local or remote servers
  - Independent of the storage engine
  - Written in text format
  - Portable
  - Excellent copy/move strategy
  - Good for small exports but not a full backup solution
- Basic usage:

```
mysqldump --user=username --password=password db_name > backup.file
```



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

## Storage Location

For SQL-formatted dump files that contain `CREATE TABLE` and `INSERT` statements, the server sends the table contents to `mysqldump`, which writes the output on the client host.

## Ensuring Data Consistency with mysqldump

Ensuring consistency:

- **--master-data** option alone
  - Locks all tables during backup with FLUSH TABLES WITH READ LOCK
  - Records the binlog position as a CHANGE MASTER TO statement in the backup file
    - --master-data=2 records the position as a comment
- **--master-data** and **--single-transaction** options used together
  - Does not lock tables – only InnoDB table consistency is guaranteed
  - Acquires a global lock at the beginning of the backup operation to obtain a consistent binary log position
- **--lock-all-tables**
  - Satisfies consistency by locking all tables for the duration of the whole dump
- **--flush-logs**
  - Starts a new binary log



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

The **--single-transaction** option allows non-blocking data consistency by using a repeatable read isolation level transaction to read all the InnoDB data.

## mysqldump Options for Creating Objects

- **--no-create-db**
  - Suppresses the CREATE DATABASE statements
- **--no-create-info**
  - Suppresses the CREATE TABLE statements
- **--no-data**
  - Creates the database and table structures but does not dump the data
- **--no-tablespaces**
  - Tells the MySQL server not to write any CREATE LOGFILE GROUP or CREATE TABLESPACE statements to the output
- **--quick**
  - Retrieves single rows from a table, without buffering sets of rows



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

## mysqldump Options for Dropping Objects

- **--add-drop-database**
  - Adds a `DROP DATABASE` statement before each `CREATE DATABASE` statement
- **--add-drop-table**
  - Adds a `DROP TABLE` statement before each `CREATE TABLE` statement



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

## mysqldump General Options

- Stored routines, events, and triggers:
  - **--routines**
    - Dumps stored routines (procedures and functions) from the dumped databases
  - **--events**
    - Dumps events created for the Event Scheduler
  - **--triggers**
    - Dumps triggers for each dumped table
- Top options in one option (**--opt**)
  - Shorthand for the most common options to create an efficient and complete backup file
    - Includes the `--add-drop-table`, `--add-locks`, `--create-options`, `--disable-keys`, `--extended-insert`, `--lock-tables`, `--quick`, and `--set-charset` options.



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

MySQL 8.0 stores stored routines and scheduler events in the data dictionary instead of tables as in previous versions of MySQL Server. Therefore, a backup of the `mysql` database does not contain the stored procedures and events. You must explicitly specify the `--routines` and `--events` options in `mysqldump` to back them up.

The `--triggers` and `--opt` options are enabled by default; use `--skip-triggers` and `--skip-opt` to disable them.

## Restoring mysqldump Backups

- Reload mysqldump backups with mysql:

```
mysql --login-path=login-path database < backup_file.sql
```

- You must name the database if the backup file does not contain the USE statements that specify the database.
- If you run mysqldump with --database or --all-databases options:
  - The dump file contains appropriate USE db\_name statements
  - You do not need to specify the target database name when reloading from that dump file

- Copy from one database to another:

```
mysqldump -uuser -ppassword orig-db | mysql -uuser -ppassword copy-db
```



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

## Copying from One Database to Another

You can use mysqldump output to restore tables or databases and to copy them. mysql can read from a pipe, so the use of mysqldump and mysql can be combined into a single command that copies tables from one database to another. The pipe technique also can be used to copy databases or tables over the network to another server.

## Using mysqlimport

- If you invoke `mysqldump` with the `--tab` option, it produces tab-delimited data files:
  - A `.sql` file that contains `CREATE TABLE` statements
  - A `.txt` text file that contains the table data
- To reload the table:
  1. Change to the backup directory.
  2. Process the `.sql` file by using `mysql`.
  3. Load the `.txt` file by using `mysqlimport`.

```
1 cd backup_dir
2 mysql -uuser -ppassword database < table.sql
3 mysqlimport -uuser -ppassword database table.txt
```



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

### **mysqlimport**

If you combine the `--tab` option with options such as `--fields-terminated-by` and `--fields-enclosed-by`, which perform format control, specify the same format-control options with `mysqlimport` so that it knows how to interpret the data files.

## Privileges Required for mysqldump

You must have the following privileges to use `mysqldump`:

- `SELECT` for dumped tables
- `SHOW VIEW` for dumped views
- `TRIGGER` for dumped triggers
- `LOCK TABLES` (unless you use the `--single-transaction` option)
- Other options might require extra privileges. For example:
  - To use the `--flush-logs` or `--master-data` options, you must have the `RELOAD` privilege.
  - To create a tab-delimited output with the `--tab` option, you must have the `FILE` privilege.
  - To use the `--routines` option to back up stored functions and procedures, you must have the global `SELECT` privilege.



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

## Privileges Required for Reloading Dump Files

To reload a dump file, you must have the following privileges:

- **CREATE** on each of the dumped objects
- **INSERT** on each of the dumped tables
- **ALTER** on the database, if the `mysqldump` output includes statements that change the database collation to preserve character encodings



ORACLE®

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

## `mysqlpump`

`mysqlpump` is very similar to `mysqldump`, but with the following enhancements:

- Provides better performance than `mysqldump` by extracting data in parallel threads
  - `mysqlpump` divides the dump process into several subtasks and then adds these subtasks to a multithreaded queue.
  - This queue is then processed by  $N$  threads (two by default).
- Enables better control over which database objects to dump on each thread
- Dumps users as `CREATE USER/GRANT` statements instead of as `INSERTS` into the `mysql` system database
- Enables compressed output
- Displays a progress indicator
- Provides faster secondary index reloading for InnoDB tables



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

Faster secondary index reloading for InnoDB tables is achieved by adding indexes after the rows are inserted.

## Specifying Objects to Back Up with mysqlpump

The mysqlpump utility provides many options for specifying which database objects to back up, for example:

- Back up all databases (default):

```
mysqlpump -uuser -ppassword > full_backup.sql
```

- Back up *only* the employees and world databases:

```
mysqlpump -uuser -ppassword
--databases employees world > emp_world_backup.sql
```

- Dump all databases with names that start with “db.”

```
mysqlpump -uuser -ppassword
--include-databases=db% --result-file=all_db_backup.sql
```

- Back up everything *except* tables named t1.

```
mysqlpump -uuser -ppassword --exclude-tables=t1
--result-file=partial_backup.sql
```



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

Unlike mysqldump, the default behavior of mysqlpump is to back up *all* databases. This option is implicit if you execute mysqlpump without specifying which schemas to include.

By default, mysqlpump does not dump the following internal databases: PERFORMANCE\_SCHEMA, INFORMATION\_SCHEMA, sys, and ndbinfo. To dump any of these, name them explicitly with the --databases or --include-databases option.

mysqlpump dumps user accounts in logical form using CREATE USER and GRANT statements (for example, when you use the --include-users or --users option). For this reason, dumps of the mysql system database do not by default include the grant tables that contain user definitions: user, db, tables\_priv, columns\_priv, procs\_priv, or proxies\_priv. To dump any of the grant tables, name the mysql database followed by the table names.

## Parallel Processing with mysqlpump

You can configure the number of threads `mysqlpump` uses with the `--default-parallelism` and `--parallel-schemas` options. For example:

- Back up all databases with four threads:

```
mysqlpump --uuser -ppassword
--default-parallelism=4 > full_backup.sql
```

- Create two queues: one to process `db1` and `db2` and the other to process `db3` and `db4`. Use five threads on the first queue, two on the second, and three on the default queue for all other schemas:

```
mysqlpump --uuser -ppassword
--parallel-schemas=5:db1,db2
--parallel-schemas=2:db3,db4
--default-parallelism=3 > full_backup.sql
```



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

### Note

The default number of threads (the value of `--default-parallelism`) is two.

## Quiz



`mysqldump` and `mysqlpump` are useful for small exports but not as a full backup solution.

- a. True
- b. False



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

**Answer: a**

## Topics

- MySQL backup tools
- Physical backups using MySQL Enterprise Backup
- Logical backups using `mysqldump` and `mysqlpump`
- Other physical backup methods
- Replication-based backups
- Recover database using binary logs



ORACLE®

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

## Physical InnoDB Backups: Overview

- A physical (binary) backup operation makes a complete InnoDB backup.
  - Contains a backup of all InnoDB tables
  - Makes exact copies of all tablespace files
- All InnoDB tables in all databases must be backed up together.
  - InnoDB maintains some information centrally in the system tablespace.
  - Other InnoDB tablespaces contain table data that is dependent on InnoDB's data dictionary in the system tablespace.
- You must shut down the MySQL server for the duration of the file copy to ensure consistency across all tablespaces.
  - This is a “cold” backup.



ORACLE®

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

## Portability of Physical Backups

- Binary portability
  - Binary database files can be copied from one MySQL server to another.
  - Binary portability is useful when taking a physical backup made on one machine to use on a second machine that has a different architecture.
    - For example, you can copy databases from one MySQL server to another by copying the binary database files.
- Storage engine portability
  - InnoDB:
    - All tablespace and log files for the database can be copied directly.
      - Example: You can copy tablespace files directly from a MySQL server on one machine to a MySQL server on another machine, and the second server accesses the tablespace.
      - The database directory name must be the same on the source and destination systems.
  - MyISAM and Archive:
    - All files for an individual table can be directly copied and the metadata can be imported from a .sdi file.



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

### File name compatibility for Windows:

- The MySQL server internally stores lowercase database and table names on Windows systems.
- For case-sensitive filesystems, use an option file statement:

`lower_case_table_names=1`

In MySQL 8.0, it is prohibited to start the server with a `lower_case_table_names` setting that is different from the setting used when the server was initialized. You must set the option before you initialized the data directory and cannot change the setting after that.

## Physical InnoDB Backup Procedure

1. Perform a slow (clean) shutdown of the server.
  - Requires `innodb_fast_shutdown=0` (The default value is 1.)
  - Allows additional InnoDB flushing operations to complete before shutdown
  - Shutdown takes longer; startup is faster.
2. Make a copy of *all* InnoDB data, log, and configuration files:
  - Data files: `ibdata` and `*.ibd`
  - Redo log files: `ib_logfile*`
  - Undo tablespace files: `undo_*`
  - All configuration files that the server applies, such as `my.cnf`
3. Restart the server.



ORACLE®

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

## Recovering from Physical InnoDB Backups

- To recover InnoDB tables by using a physical backup:
  1. Stop the server
  2. Replace all the components of which copies were made during the backup procedure
  3. Restart the server
- InnoDB stores table metadata in the shared tablespace. You must:
  - Copy the shared tablespace and per-table tablespace files as a group
    - This replaces the target system tablespace.
  - Copy the corresponding redo log files and undo tablespace files



ORACLE®

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

## Using Transportable Tablespaces for Backup

When you back up InnoDB tables by using transportable tablespaces:

- You can perform partial backups
  - Per-table backups or multiple tables within a business function
- You do not need to load those tables in another server
  - If you want to read the backed-up copies for analytics or other purposes, ensure InnoDB is set to read only on that server.
  - You must also ensure that the original server and the read-only server have matching InnoDB page sizes and row formats.
  - You must not change the tables by using them in another server.
  - If you change the tables, the backup no longer has integrity.
- You can restore partial backups per table or per business function



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

## Transportable Tablespaces: Copying a Table to Another Instance

1. On the source instance, execute FLUSH TABLES...FOR EXPORT to quiesce the table and create the .cfg metadata file:

```
mysql> FLUSH TABLES db.table FOR EXPORT;
```

- The .cfg file is created in the InnoDB data directory.

2. Copy the .ibd and .cfg files from the source instance to the destination instance.
3. On the source instance, release the locks by executing UNLOCK TABLES.
4. On the destination instance, create a table with the same structure as that on the source instance.

```
mysql> ALTER TABLE db.table DISCARD TABLESPACE;
```

5. On the destination instance, discard the existing tablespace:
6. On the destination instance, import the tablespace:

```
mysql> ALTER TABLE db.table IMPORT TABLESPACE;
```



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

If you are copying an InnoDB partitioned table, then there will be a separate tablespace (.ibd file) and .cfg file for each partition. You must discard each of the tablespaces on the destination instance, copy all the .ibd and .cfg files, and import all the tablespaces.

## Physical MyISAM and ARCHIVE Backups

- Physical MyISAM and ARCHIVE backups comprise the files that MySQL uses to represent the tables.
  - For MyISAM, these are the `.sdi`, `.MYD`, and `.MYI` files.
  - For ARCHIVE tables, these are the `.sdi` and `.ARZ` files.
- During this copy operation, other programs (including the server) must not modify the files being copied.
  - To avoid server interaction problems, lock and flush the tables or stop the server during the copy operation.



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

### Note

Locking tables instead of shutting down the server works on Linux systems. On Windows, file-locking behavior is such that it might not be able to copy table files for tables that are locked by the server. In that case, stop the server before copying table files.

## Physical MyISAM and ARCHIVE Backup Procedure

1. While the server is running, lock the table to be copied:

```
mysql> USE world_myisam
mysql> FLUSH TABLES city WITH READ LOCK;
```

2. Perform a filesystem copy:

```
cp datadir/world_myisam/city.* /backupdir/
```

3. Start a new binary log file:

```
mysql> FLUSH LOGS;
```

- The new binary log file contains all statements that change data after the backup (and any subsequent binary log files).

4. Release the lock after the filesystem copy:

```
mysql> UNLOCK TABLES;
```



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

If you do not want to flush the logs, you can record the current log coordinates with the `SHOW MASTER STATUS` command, which returns the current binary log file name and position.

## Recovering from Physical MyISAM or Archive Backups

To recover a MyISAM or ARCHIVE table from a physical backup:

1. Drop the existing table if it exists
2. Copy the backup data file into the appropriate database directory
3. Copy the backup .sdi file into a temporary directory on the server host
4. Run the `IMPORT TABLE` statement specifying the path to the .sdi file to load the table into the database

```
mysql> IMPORT TABLE FROM 'sdi_file_path';
```



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

Note that the .sdi file contains the schema name and table name. You may need to edit the file if you are restoring it to a different database or table name.

The `IMPORT TABLE` statement loads the data dictionary into the MySQL server.

## LVM Snapshots

Logical Volume Manager (LVM):

- Manages storage areas on physical media that are more flexible than disk partitions
  - Physical volumes
  - Logical volumes
  - Volume groups
- Uses a mechanism known as “copy-on-write” to create snapshots
  - The snapshot behaves as if it is an instantaneous copy of the filesystem at the time you create the snapshot.
  - Internally, the snapshot records the original versions of disk blocks that change after you create the snapshot.
  - When you access the snapshot, LVM returns either unchanged blocks from the filesystem or the original copy of changed blocks.



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

### Copy-On-Write Snapshots

When you read files from a newly created snapshot, LVM reads those files from the original volume. As the original volume changes, LVM copies data to the snapshot immediately before it changes on the original so that any data that has changed since taking the snapshot is stored in its original form in the snapshot. The result is that when you read files from the snapshot, it delivers the version of data existing at the instant the snapshot was created.

Because a snapshot is almost instantaneous, you can assume that no changes to the underlying data take place during the snapshot. This makes snapshots very useful for backing up InnoDB databases without shutting the server down if all the files are stored in a single logical volume that can be snapshot as a single snapshot. If the files span across multiple logical volumes, you must shut down the MySQL database server and snapshot all the volumes containing MySQL data files before starting up the server again.

## LVM Backup Procedure

- Perform physical backups using LVM snapshots when:
  - The host supports LVM
    - Example: Linux supports LVM2.
  - The filesystem containing the MySQL data directory is on a logical volume
- Backup procedure:
  1. Shut down the MySQL server or lock the tables if needed.
    - Shutdown is needed if InnoDB files span across multiple volumes or if you want to take a consistent backup.
    - Use FLUSH TABLES WITH READ LOCK if you are backing up non-InnoDB tables.
  2. Take a snapshot of the logical volume containing MySQL's data directory.
  3. Restart the MySQL server or unlock the tables if needed.
  4. Perform a physical backup from the snapshot.
  5. Remove the snapshot.



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

### Examples of LVM commands:

- Create a snapshot called `lv_datadirbackup` from the `/dev/VG_MYSQL/lv_datadir` logical volume with a reserved size of 2GB.
  - `lvcreate -s -n lv_datadirbackup -L 2G /dev/VG_MYSQL/lv_datadir`
- Mount the snapshot and back up the files using tar.
  - `mkdir /mnt/snap`
  - `mount /dev/VG_MYSQL/lv_datadirbackup /mnt/snap`
  - `tar -cvzf backup.tar /mnt/snap`
  - `umount /mnt/snap`
- Remove the snapshot called `lv_datadirbackup` from the volume `VG_MYSQL`.
  - `lvremove VG_MYSQL/lv_datadirbackup`

# Backing Up Log and Status Files

- Binary log files
- Option files used by the server (`my.cnf` and `my.ini` files)
- Replication files:
  - `master.info`
  - `relay-log.info`
- Replication slave data files
  - `SQL_LOAD-*`
- MySQL binaries and libraries
- Strategies:
  - Static files: Backed up with normal system tools with the server running
  - Dynamic files: Backed up with normal system tools with the server stopped



ORACLE

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

## Binary Log Files

Binary logs store updates that have been made after the backup was made.

## Option Files Used by the Server (`my.cnf` and `my.ini` files)

These files contain configuration information that must be restored after a crash.

## Replication Files

Replication slave servers create a `master.info` file that contains information needed for connecting to the master server and a `relay-log.info` file that indicates the current progress in processing the relay logs.

## Replication Slave Data Files

Replication slaves create data files for processing `LOAD DATA INFILE` statements. These files are located in the directory named by the `slave_load_tmpdir` system variable, which you can set by starting the server with the `--slave-load-tmpdir` option. If you do not set `slave_load_tmpdir`, the value of the `tmpdir` system variable applies. To safeguard replication slave data, back up the files beginning with `SQL_LOAD-`.

## Topics

- MySQL backup tools
- Physical backups using MySQL Enterprise Backup
- Logical backups using `mysqldump` and `mysqlpump`
- Other physical backup methods
- Replication-based backups
- Recover database using binary logs



ORACLE®

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

## Replication as an Aid to Backup

Replicate to another server to avoid backup overhead.

- Use the slave server to make backups instead of backing up from the master.
- By using a slave server for backups:
  - Applications that use the master are not slowed down by table locks or flushes
  - The backup procedure does not add processing load on the master
  - Backup files do not require additional hard disk space or processing on the master



ORACLE®

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

## Backing Up from a Replication Slave

**1.** Stop the slave to make a backup:

- STOP SLAVE SQL\_THREAD
  - Flush the tables if you want to make a logical backup.
  - Alternatively, stop the server to make a physical file copy.

**2.** Back up the slave's databases:

- Use mysqlimport or mysqlbackup to back up the contents of a running server.
- Use system tools to copy physical files if you stop the server.
  - Make sure the slave\_open\_temp\_tables status variable is 0 before you stop the slave server.

**3.** Start the slave:

- Start the server if it is stopped.
- START SLAVE SQL\_THREAD



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

If you stop the slave server with temporary tables that are opened for use in updates that have not yet been executed, the temporary tables needed by those updates are no longer available when the slave is restarted.

## Backing Up from Multiple Sources to a Single Server

- Multisource replication can be used to:
  - Back up multiple servers to a single server
  - Merge table shards
  - Consolidate data from multiple servers to a single server
- The slave creates a replication channel for each master from which it receives transactions.
  - Use the `CHANGE MASTER TO ... FOR CHANNEL channelname` syntax:
- Replicate from all channels concurrently or start and stop individual channels:

```
CHANGE MASTER TO ...
MASTER_LOG_FILE='binlog.000006',
MASTER_LOG_POS=143 FOR CHANNEL 'shard-1';
```

```
START SLAVE IO_THREAD FOR CHANNEL 'shard-1';
```



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

If you are consolidating multiple shards as part of a backup strategy, you might enable the slave I/O threads only at certain times of the day to avoid placing undue stress on the shard master servers during peak times. However, bear in mind that all relevant events on the master server must replicate, and that might take quite a lot of time at the slave server.

## Topics

- MySQL backup tools
- Physical backups using MySQL Enterprise Backup
- Logical backups using `mysqldump` and `mysqlpump`
- Other physical backup methods
- Replication-based backups
- Recover database using binary logs



ORACLE®

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

## Processing Binary Log Contents

1. Determine which logs were written after a backup was made.
2. Convert the contents with `mysqlbinlog`:

```
mysqlbinlog binlog.000050 binlog.000051 binlog.000052 | mysql
```

- Process all binlogs with one command.

3. Remove the binary logs that are no longer useful and that take up space on the server host:
  - Use the `--delete-master-logs` option when you create a backup using `mysqldump`.
  - Execute `PURGE BINARY LOGS` to remove files or events that are no longer useful.
    - Do not remove log files or events that have not yet been replicated or backed up.

```
PURGE BINARY LOGS TO 'binlog.000048' ;
```



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

After you restore the binary backup files or reload the text backup files, finish the recovery operations by reprocessing the data changes that are recorded in the server's binary logs.

To do this, you must determine which logs were written after the backup was made. The contents of these binary logs then need to be converted to text SQL statements with the `mysqlbinlog` program to process the resulting statements with `mysql`.

## Selective Binary Log Processing

- Restore partial binlogs:

- --start-position
    - Specifies the log position of the first logged statement that you want to extract
  - --stop-position
    - Specifies the log position to stop extracting the log contents

```
mysqlbinlog --start-position=23456 binlog.000004 | mysql
```

- Select only those events that occur after selecting a specific default database with --database:

```
mysqlbinlog --database=sales binlog.000043 | mysql
```

- Restore to a different database with --rewrite-db:

```
mysqlbinlog --rewrite-db='sales->sales_03' binlog.000082 | mysql
```



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

mysqlbinlog also supports the --start-datetime and --stop-datetime options. These options use the event timestamps (the event start time) recorded in the binary logs. As the events binary logs are ordered based on the commit sequence, the event timestamp may not be in sorted order. It is recommended to inspect the binary logs to obtain the position instead of using the date time options.

## Point-in-Time Recovery

You might want to apply only portions of the binary log *before* or *after* a point in time:

- **Before:**
  - If you know that somebody accidentally executed a destructive statement and you want to restore changes up to that point
  - If you are restoring changes up to the last moment of a particular day to an analytics server
- **After:**
  - If you are using a binary log to apply changes after a full backup or after a flushed binary log that you have backed up, you can apply the log after that point.



ORACLE®

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

## Configuring MySQL for Restore Operations

- MySQL client programs might fail due to buffer overflows.
  - During logical backup using `mysqldump`
  - During dump file restore using `mysql`
  - During binary log restore operations using `mysqlbinlog` and `mysql`
- Configure the following variables:
  - `--max-allowed-packet=256M`
    - Default value: 64M
    - Ensure that the value is large enough during backup and restore operations so that you do not exceed MySQL's packet size.



ORACLE®

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

## Quiz



Which command converts the contents of binary logs?

- a. binconvert
- b. mysql
- c. mysqlbinlog
- d. mysqldump



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

**Answer: c**

## Summary



In this lesson, you should have learned how to:

- Use MySQL Enterprise Backup to perform consistent backups
- Use the `mysqldump` and `mysqlpump` utilities to perform logical backups
- Explain when and how to use physical file backups
- Back up from a replication slave
- Use binary logs to recover the database



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

## Practices

- 11-1: Backing up with MySQL Enterprise Backup
- 11-2: Backing up with `mysqldump` and `mysqlpump`
- 11-3: Restoring the Database Using the Binary Log



ORACLE®

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



# 12

# Configuring a Replication Topology



ORACLE®

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

## Objectives



After completing this lesson, you should be able to:

- Describe MySQL replication
- Explain the role of replication in high availability and scalability
- Set up a MySQL replication environment
- Design advanced replication topologies
- Clone MySQL data



ORACLE®

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

## Topics

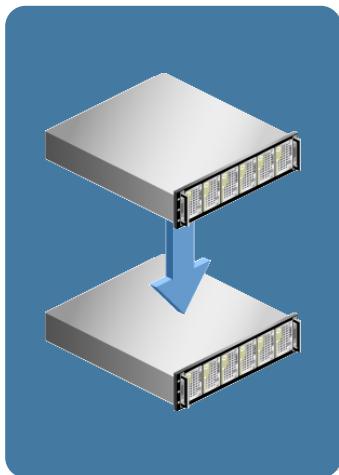
- Replication overview
  - Replication conflicts
  - When to use replication
  - Configuring replication
  - Binary and Replication Logs
  - Types of Replication
  - Cloning MySQL Data



ORACLE®

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

## MySQL Replication



Replication is a feature of MySQL that enables servers to copy changes from one instance to another.

- The **master** writes all data and structural changes to the binary log.
  - The binary log format is statement based, row based, or mixed.
- The **slave** requests the binary log from the master and applies its contents locally.
  - It keeps track of the status of all the received and applied events so that it can resume from where it has stopped after a server restart or network failure.

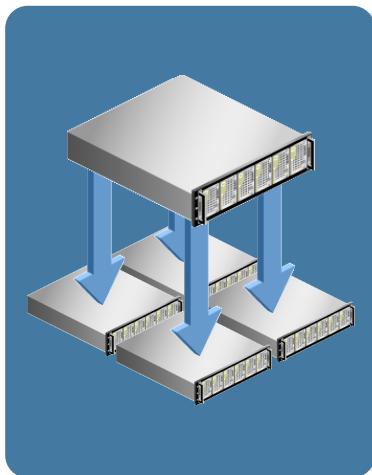
ORACLE

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

### Note

- **Binary log formats:** MySQL supports statement-based, row-based, and mixed format logging as described in the “Binary Log Formats” slide later in the lesson.
- **Network Outages:** Replication in MySQL survives a network outage. Each slave keeps track of how much of the log it has processed and resumes processing automatically when network connectivity is restored. This behavior is automatic and requires no special configuration.

## Replication Masters and Slaves



The master/slave relationship is usually one-to-many:

- Each slave reads logs from one master.
- A master can ship logs to many slaves.

MySQL supports multi-source replication:

- A slave can replicate from multiple masters.

ORACLE

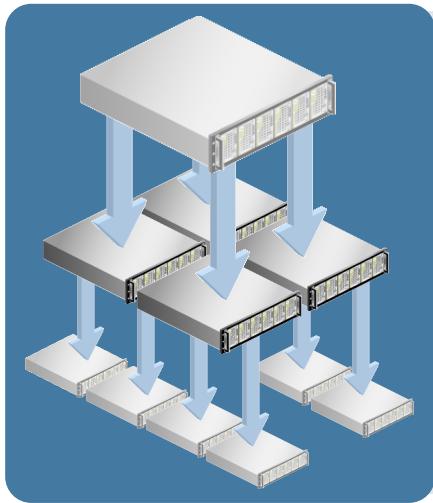
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

### Number of Slaves

There is no limit on how many slaves a single master can have. However, each additional slave uses a small amount of resources on the master, so you should carefully consider the benefit of each slave in production setups. The optimal number of slaves for a master in a given environment depends on several factors: size of schema, number of writes, relative performance of master and slaves, and factors such as CPU and memory availability. A general guideline is to limit the number of slaves per master to no more than 30.

**Note:** Usually a slave replicates only from a single master. However, a slave can replicate from multiple master servers in a Multi-source replication scenario. Configuring Multi-source replication is described in the slide titled “Multi-Source Replication.”

## Relay Slaves



- A relay slave is a replication slave that acts as a replication master to another slave.
- Changes propagate to further slaves.

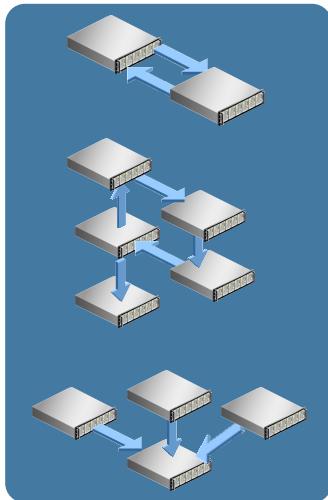
ORACLE®

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

### Relay Slave

A slave can act as a master to another slave. Changes that occur at the top-most master are requested and applied by its immediate slaves, which relay the changes down to their slaves and so on until replication reaches the end of the chain. This enables updates to propagate through multiple levels of replication, allowing for topologies that are more complex. Each additional level adds more propagation delays into the system, so a shallower setup suffers from less replication lag than a deeper one.

## Complex Topologies



More complex topologies are possible:

- A *bi-directional* topology has two master servers, each a slave of the other.
- A *circular* topology has any number of servers.
  - Each is both a master and a slave of another master.
  - Changes on any master replicate to all servers.
  - Not every slave must be a master.
- *Multi-source replication* allows one slave to receive transactions from more than one master.

ORACLE®

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

### Note

Standard MySQL replication does not perform conflict resolution.

## Quiz



A *relay slave* is:

- a. A server that logs any transactions that cannot be applied to other slaves
- b. A server that resolves conflicts by coordinating changes from multiple masters
- c. A slave server that acts as a master to another slave
- d. The first slave in a circular replication topology



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

**Answer: c**

## Topics

- Replication overview
- Replication conflicts
- When to use replication
- Configuring replication
- Binary and Replication Logs
- Types of Replication
- Cloning MySQL Data



ORACLE®

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

## Replication Conflicts

Conflicts are possible in replication topologies with multiple masters.

- If two clients write to the same row on two masters at approximately the same time, you cannot predict the row's final value at slave servers.
- The final value depends on the event ordering at the relay slave.
  - In hierarchical replication, the row's final value on slaves is implied by the hierarchy:
    - Slaves do not have the same value as masters if some intermediate master changed the row.
  - In circular replication, the row's final value is inconsistent across servers in case of a conflict.
    - The value depends on the order in which each master applies the event.
- Standard MySQL replication does not perform conflict resolution. However, MySQL Group Replication can detect and prevent conflicts.



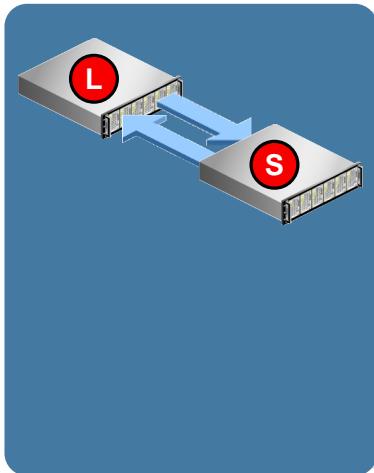
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

## Conflict Resolution

In a typical configuration, clients write changes only to the master but read changes from any server. In an environment where servers allow concurrent updates on similar data, the eventual state of the data on multiple servers might become inconsistent. It is the responsibility of the application to prevent or manage conflicting operations. MySQL replication does not perform conflict resolution.

Conflicts can occur in any topology that includes more than one master. This includes simple hierarchies such as that shown in the slide titled “Relay Slaves,” if a relay slave accepts changes from clients. Conflicts are particularly prevalent in circular topologies.

## Replication Conflicts: Example Scenario with No Conflict



- “Luxury Brands” increases the price of luxury products by 20%. **L**
- “Special Events” reduces the prices of products over \$500 by \$50. **S**
- One luxury product costs \$520 before these changes.
- If the operations perform in the order **L** and then **S**, the final result is \$574 on both servers.
- If the operations perform in the order **S** and then **L**, the final result is \$564 on both servers.

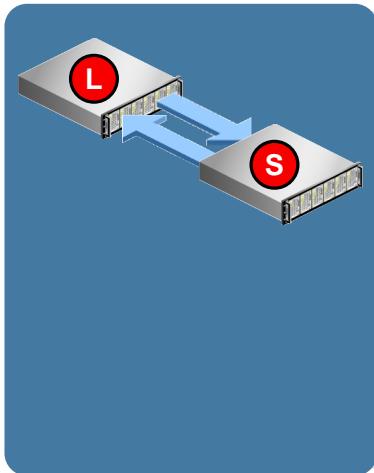
ORACLE®

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

For example, consider an e-commerce company that implements replication using a circular topology, in which two of the servers deal with applications in the “Luxury Brands” and “Special Events” teams, respectively. Assume that the applications do not manage conflicting operations, and the following events occur:

- The “Luxury Brands” team increases the price of luxury products by 20%.
- The “Special Events” team reduces the prices of all products over \$500 by \$50 because of an upcoming special holiday.
- One product costing \$520 falls into both categories and has its value updated by both of the preceding operations.

## Replication Conflicts: Example Scenario with Conflict



- If both operations occur at approximately the same time, the following operations occur:
  - **L** Increases the product's price by 20%, from \$520 to \$624
  - **S** Reduces the product's price by \$50, from \$520 to \$470
- If these operations occur at the same time, each of the changes replicate to the other server resulting in a conflict.
  - The “Luxury Brands” server assumes a final value for that product of \$574. **L**
  - The “Special Events” server assumes a final value of \$564. **S**

ORACLE

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

The eventual price of the product on each server depends on the order in which each server performs the operations.

- Other servers in the replication environment assume final values depending on the order in which they applied the operations.

Similarly, conflicts occur if the “Luxury Brands” team adds a new product that has not replicated fully by the time the “Special Events” team makes its changes or if two teams add a product with the same primary key on different servers.

As MySQL servers cannot detect and resolve replication conflicts, the application has to handle all the replication conflicts that may occur.

**Note for MySQL Cluster users:** MySQL Cluster uses a form of replication internally that differs in some ways from replication in the MySQL server and offers conflict detection (and optional resolution).

## Topics

- Replication overview
- Replication conflicts
- When to use replication
- Configuring replication
- Binary and Replication Logs
- Types of Replication
- Cloning MySQL Data



ORACLE®

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

## Replication Use Cases

Common uses for replication:

- **Horizontal scale-out:** Distribute the querying (read) workload across multiple slaves
- **Business intelligence and analytics:** Run expensive reports and analytics on a slave, letting the master focus on production applications
- **Geographic data distribution:** Serve local users with a local application and replicate business intelligence data to corporate servers
- **High availability:** Provide redundancy between multiple servers and facilitate controlled switchovers or rolling upgrades



ORACLE®

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

## Replication for Horizontal Scale-Out

- Distribute the client query load across multiple servers:
  - Clients send write operations to the replication master.
  - Clients distribute read operations between replication slaves.
    - Use a load balancer or connectors that provide load balancing.
- Conflicts cannot occur because only one server accepts write operations.
  - Inconsistencies are possible if an application writes a value and immediately tries to read it again.
- Scale-out is not transparent. You must modify clients to:
  - Write to the master
  - Read from a load-balanced set of slaves
  - Handle potential inconsistencies that occur due to the delay in replicating from master to slaves



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

**Horizontal scale-out:** The most popular reason to implement replication is to distribute the querying workload across one or more slave servers to improve the performance of read operations throughout the application and write operations on the master server by reducing its read workload.

## Replication for Business Intelligence and Analytics

- Business intelligence workloads often have different performance requirements to transactional data.
  - High throughput due to reading many rows from multiple tables
  - Long-running queries with aggregation and summarization
- Avoid locking your transactional workload by replicating business intelligence data to a dedicated analytics slave.
  - Use multiple slaves for different business units.
  - Configure indexes and storage engines on each slave to optimize for different analytics requirements.
  - Use replication filters or the BLACKHOLE storage engine to avoid replicating table data that you do not need to store on each slave.



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

**Business intelligence and analytics:** Business intelligence reports and analytical processing can be resource intensive and can take considerable time to execute. In a replicated environment, you can run such queries on a slave so that the master can continue to service the production workload without being affected by long-running and I/O-intensive reports.

### Note

- Replication filters are described in the slide titled “Replication Filtering Rules.”
- Replicating with the BLACKHOLE storage engine is described in the slide titled “Replicating with the BLACKHOLE Storage Engine.”

## Replication for Geographic Data Distribution

- Store geographically relevant data on servers that are located near their primary users.
  - Use scaled-out replication at each location to increase the potential throughput for larger local user bases.
- Replicate some or all of this data to other locations.
  - Replicate location-independent data to all locations.
    - Users, products, categories, sales records
  - Replicate summaries to central corporate servers.
    - Sales aggregates, campaign outcomes, inventory data



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

**Geographic data distribution:** Companies with a distributed geographic presence can benefit from replication by having servers in each region that process local data and replicate that data across the organization. This provides the performance and administrative benefits of geographic proximity to customers and the workforce, while also enabling corporate awareness of data throughout the company.

**Note:** Multi-source replication is ideal for geographically distributed servers. See the slide titled “Multi-Source Replication” later in this lesson.

## Replicating with the BLACKHOLE Storage Engine

- The BLACKHOLE storage engine silently discards all data changes with no warnings.
  - It accepts DML requests to insert or update data, even though there cannot be any matching data.
  - It does not have any further intelligence or function.
- The binary log continues to record those changes successfully.
- Use BLACKHOLE for tables on a relay slave when it replicates all changes to further slaves, but does not itself need to store data in those tables.



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

You can use replication filters to prevent a server from replicating some data, but if you need to replicate that data to further slaves even though you are not storing it locally, use the BLACKHOLE storage engine to avoid saving that data locally.

For example, if you have a relay slave that is used solely for executing frequent long-running business intelligence reports on a small number of tables, you can configure all other replicated tables to use BLACKHOLE so that the server does not store data it does not need, while it replicates all changes to its slaves.

**Note:** Replication filters are described in the slide titled “Replication Filtering Rules.”

## Replication for High Availability

Replication allows various high-availability use cases.

- **Controlled switchover:** Use a replica to act in place of a production server during hardware or system upgrades.
- **Server redundancy:** Perform a failover to a replica server in case of a system failure.
- **Online schema changes:** Perform a rolling upgrade in an environment with several servers to avoid an overall system outage.
- **Software upgrades:** Replicate across different versions of MySQL during an environment upgrade.
  - Slaves must run a later version than the master.
  - Queries issued during the upgrade must be supported by all versions used during the upgrade process.



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

## Topics

- Replication overview
- Replication conflicts
- When to use replication
- **Configuring replication**
- Binary and Replication Logs
- Types of Replication
- Cloning MySQL Data



ORACLE®

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

## Configuring Replication

1. Draw a replication topology diagram.
2. Identify all servers that participate in replication.
  - Note that each replication master that replicates from another master is also a replication slave.
3. Configure a unique **server-id** for each server.
  - An unsigned 32-bit integer with a default value of 1
  - Any master or slave server with a **server-id** of 0 refuses to replicate with other servers.
4. Configure each replication master.
5. Configure each replication slave to connect to its master.
6. Start replication on each replication slave with **START SLAVE**.



ORACLE®

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

## Configuring Replication Masters

- Enable TCP/IP networking.
    - Replication cannot use UNIX socket files.
  - Enable the binary log.
    - During replication, each master sends its log contents to each slave.
  - Create a user with the REPLICATION SLAVE privilege.

```
CREATE USER user@slave_hostname IDENTIFIED BY 'password';
GRANT REPLICATION SLAVE ON *.* TO user@slave_hostname;
```
  - Each slave must connect to a master to replicate from it.
  - On masters with multiple slaves, specify wildcards in the host name to match all slaves or create multiple users.
- Back up the master database as a starting point for the slave.
    - Record the log coordinates if you are not using GTIDs.
      - Use the --master-data option if you are using mysqldump.



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

The caching\_sha2\_password authentication plugin is the default for new users created from MySQL 8.0. To connect to the replication master using a user account that authenticates with the caching\_sha2\_password plugin, you must either set up a secure connection as described in <https://dev.mysql.com/doc/refman/8.0/en/replication-solutions-encrypted-connections.html>, or enable the unencrypted connection to support password exchange using an RSA key pair.

**Note:** GTIDs refer to Global Transaction Identifiers. This lesson covers GTIDs in the slide titled “Global Transaction Identifiers (GTIDs).”

If you are using GTID, the executed GTID sets captured in the backup are required to configure the slave servers. You can use the --set-gtid-purged option in mysqldump to control the SET @@GLOBAL.GTID\_PURGED command in the generated SQL script.

## Configuring Replication Slaves

- Restore the backup from the master.
  - Verify that the `gtid_purged` variable is set if you are using GTIDs.
- Issue a `CHANGE MASTER TO` statement on each slave with the:
  - Network location of the master
    - `MASTER_HOST` and `MASTER_PORT` values
    - Optionally, use `MASTER_SSL` and related options to encrypt network traffic between masters and slaves during replication.
  - Replication account username and password (with the `REPLICATION SLAVE` privilege)
    - `MASTER_USER` and `MASTER_PASSWORD` values
  - Binary log coordinates from which to start replicating (if you are not using GTIDs)
    - The `MASTER_LOG_FILE` and `MASTER_LOG_POS` values store the binary log position from which the slave starts replicating.
    - Specify `MASTER_AUTO_POSITION=1` if you are using GTIDs.



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

### Note

`MASTER_SSL` and related options are available only on SSL-enabled servers.

## CHANGE MASTER TO

- Issue the `CHANGE MASTER TO...` statement on the slave to configure replication master connection details:

```
mysql> CHANGE MASTER TO
 -> MASTER_HOST = 'host_name',
 -> MASTER_PORT = port_num,
 -> MASTER_USER = 'user_name',
 -> MASTER_PASSWORD = 'password',
 -> MASTER_LOG_FILE = 'master_log_name',
 -> MASTER_LOG_POS = master_log_pos;
```

- Subsequent invocations of `CHANGE MASTER TO` retain the value of each unspecified option.
  - Changing the master's host or port also resets the log coordinates.
  - The following statement changes the password, but retains all other settings:

```
mysql> CHANGE MASTER TO MASTER_PASSWORD='newpass' ;
```



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

## Finding Log Coordinates

- Execute `SHOW MASTER STATUS` on the master immediately after performing the backup.
  - Ensure that there is no activity on the master after the backup to guarantee that the log coordinates are those of the last event before the backup.
  - Example:

```
mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----+-----+-----+-----+
| mysql-bin.000014 | 51467 | | | |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

- Use the `--master-data` option when you back up the master with `mysqldump` to find the log coordinates at the time of the backup.



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

## Global Transaction Identifiers (GTIDs)

Global Transaction Identifiers (GTIDs) uniquely identify each transaction in a replication topology.

- Each GTID is of the form `<source-uuid>:<transaction-id>`. For example:

```
0ed18583-47fd-11e2-92f3-0019b944b7f7:338
```

- A GTID set contains a range of GTIDs:

```
0ed18583-47fd-11e2-92f3-0019b944b7f7:1-338
```

- Enable GTID mode with the following options:
  - `gtid-mode=ON`: Logs a unique GTID along with each transaction
  - `enforce-gtid-consistency`: Disallows events that cannot be logged in a transactionally safe way
  - `log-slave-updates`: Records replicated events to the slave's binary log



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

The MySQL server provides some built-in functions to work with GTIDs.

- `GTID_SUBSET(set1, set2)`
  - Returns `true` if all GTIDs in `set1` are also in `set2`. Returns `false` otherwise.
- `GTID_SUBTRACT(set1, set2)`
  - Returns only those GTIDs from `set1` that are not in `set2`.
- `WAIT_FOR_EXECUTED_GTID_SET(gtid_set[, timeout])`
  - Wait until the server has applied all of the transactions whose GTIDs are contained in `gtid_set`.

**Note:** `log-slave-updates` has the default value `ON` starting from MySQL server version 8.0.3.

## Identifying the Source Server

The `server_uuid` value is:

- A universally unique identifier (UUID) stored in the `auto.cnf` file in the server's data directory
  - If the `auto.cnf` file does not exist, the server generates a new `auto.cnf` file containing a new `server_uuid` value.
  - You can inspect the `server_uuid` value by querying the system variable of the same name:

```
mysql> SELECT @@server_uuid\G
***** 1. row *****
@@server_uuid: 0ed18583-47fd-11e2-92f3-0019b944b7f7
1 row in set (#.## sec)
```

- Used in GTID to record which server the transaction has originally executed on
- Retained by all slaves in the replication chain so that the originating master for each transaction is identifiable



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

## Logging Transactions

- When each server executes a transaction, it also records that transaction's ID in the `gtid_executed` variable.
  - Contains a set of GTIDs
  - Represents all transactions from the local server, the immediate master, and any other upstream master servers

```
mysql> SELECT @@global.gtid_executed\G
***** 1. row *****
@@global.gtid_executed: bacc034d-4785-11e2-8fe9-0019b944b7f7:1-34,
c237b5cd-4785-11e2-8fe9-0019b944b7f7:1-9,
c9cec614-4785-11e2-8fea-0019b944b7f7:1-839
1 row in set (#.## sec)
```

- When the binary logs are purged, the GTID sets are removed from the binary log and stored in the `gtid_purged` system variable.
- Executing `RESET MASTER` causes both `gtid_executed` and `gtid_purged` to be set to an empty string. All the binary log files will also be deleted.



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

Whenever the binary log is rotated or the server is shut down, the server writes GTIDs for all transactions that were written into the previous binary log file into the `mysql.gtid_executed` table.

The `gtid_executed` variable is a read-only variable managed by the MySQL server while `gtid_purged` is a dynamic variable that you can configure to control your GTID replication slaves.

You may want to back up the binary log files before executing `RESET MASTER` and back up the database after executing `RESET MASTER`.

## Replication with GTIDs

Use **CHANGE MASTER TO . . .** to enable GTID replication:

- Tell the slave that transactions are identified by GTIDs:

```
CHANGE MASTER TO MASTER_AUTO_POSITION=1;
```

- You do not need to provide the log coordinates of the master (such as `MASTER_LOG_FILE` and `MASTER_LOG_POS`) because the slave sends the value of `@@global.gtid_executed` to the master. Therefore:
  - The master knows which transactions the slave has executed
  - The master sends only those transactions that the slave has not already executed
- You cannot provide `MASTER_AUTO_POSITION` and log coordinates in the same `CHANGE MASTER TO . . .` statement.



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

## Replication Filtering Rules

- Control the scope of replication by using ***replication filters***.
- Filters are server options that apply to a master or slave:
  - The master applies **binlog-\*** filters when writing the binary log.
  - The slave applies **replicate-\*** filters when reading the relay log.
- Use when different slaves in the environment serve different purposes.
  - Examples:
    - A server that is responsible for displaying web content does not require payroll or inventory data.
    - A server that provides sales data for management does not need data for web content or marketing data.



ORACLE®

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

## Applying Filtering Rules

- Choose which events to replicate, based on:
  - Database:
    - replicate-do-db, binlog-do-db
    - replicate-ignore-db, binlog-ignore-db
  - Table:
    - replicate-do-table, replicate-wild-do-table
    - replicate-ignore-table, replicate-wild-ignore-table
- Consider precedence rules when applying filters:
  - Database filters apply before table filters.
  - Table wildcard filters `*-wild-*` apply *after* those that do not use wildcards.
  - The `*-do-*` filters apply *before* the respective `*-ignore-*` filters.



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

Be careful when using multiple filters. It is very easy to make mistakes, due to the complex order in which filters are applied. Because filters control what data is replicated, such mistakes are difficult to recover from. For this reason, do not mix different types of filters.

For example, if you use `replicate-wild-*`, do not use any non-wild `replicate-*` filters.

Temporary tables are not replicated when `binlog_format` is set to `ROW` or `MIXED`. To suppress replication of a specific temporary table when `binlog_format=STATEMENT`, use `replicate-ignore-table` or `replicate-wild-ignore-table`. Other replication filters have no effect on temporary tables.

## Quiz



To use Global Transaction Identifiers in replication, you must use a CHANGE MASTER TO... statement to provide the binary log coordinates (file name and position) of the most recent transaction on the master that the slave has not yet applied.

- a. True
- b. False



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

**Answer: b**

## Topics

- Replication overview
- Replication conflicts
- When to use replication
- Configuring replication
- Binary and Replication Logs
- Types of Replication
- Cloning MySQL Data



ORACLE®

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

## Binary Log Formats

- MySQL records information in the binary log in one of three different formats:
  - Row-based logging (the default method)
  - Statement-based logging
  - Mixed logging
- Change the binary log format by starting the global or session `binlog_format` server variable:

```
SET [GLOBAL|SESSION] BINLOG_FORMAT=[row|statement|mixed|default];
```
- You cannot set the `binlog_format` variable at run time:
  - From within a stored function or a trigger
  - If the NDB storage engine is enabled
  - If the session is currently using row-based replication and has open temporary tables



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

## Row-Based Binary Logging

- Is the default binary logging format
- Records the changes to individual table rows
  - Larger log is generated when many rows are updated.
  - The values recorded in the log are exactly the same as those sent to the storage engine.
  - A primary key is required on each table to correctly identify each row.
  - Fewer DML locks are required on slaves.
- Always replays statements correctly, even if statements are non-deterministic
  - Example: CURRENT\_USER() or CONNECTION\_ID()



ORACLE®

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

Source and target tables for replication do not have to be identical. A table on the master can have more or fewer columns than the slave's copy of the table. In addition, corresponding table columns on the master and the slave can use different data types, subject to certain conditions. See <https://dev.mysql.com/doc/refman/en/replication-features-differing-tables.html> for the possible scenarios and conditions.

You can set the `slave_type_conversions` system variable to control the type conversion mode on the slave when using row-based replication.

## Statement-Based Binary Logging

- Contains the actual SQL statements
  - Can be used for auditing
  - Easier to identify statement for point-in-time recovery
- Includes both DDL (CREATE, DROP, and so on) and DML (UPDATE, DELETE, and so on) statements
- Saves disk space and network bandwidth due to the relatively small file sizes required
  - When one SQL statement modifies many rows
- Does not guarantee that non-deterministic statements replay correctly on a remote machine
  - If MySQL cannot make this guarantee, it issues a warning: Statement may not be safe to log in statement format



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

## Mixed Format Binary Logging

- Uses statement-based logging by default
- Uses row-based logging when statements are non-deterministic:
  - Calls to `UUID()`, `USER()`, `CURRENT_USER()`, `FOUND_ROWS()`, `ROWS_COUNT()`, or any user-defined functions
  - When one or more tables with `AUTO_INCREMENT` columns are updated and a trigger or stored function is invoked
  - When a statement refers to a system variable
  - Certain situations that involve views or temporary tables
- Uses row-based logging when storage engine does not support statement-based logging:
  - NDB Cluster storage engine
  - InnoDB storage engine when the isolation level is `READ COMMITTED` or `READ UNCOMMITTED`



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

## Replication Logs

Slave servers maintain information about replication events.

- Relay log set:
  - Includes relay logs and relay log index file
  - Contains a copy of binary log events from the master
- Slave status logs:
  - Contain information needed to perform replication
    - Master connection details and log coordinates
    - Relay log coordinates
  - Are stored in tables (default) or files
    - `slave_master_info` and `slave_relay_log_info` tables in the `mysql` database
    - Default file names: `master.info` and `relay-log.info`



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

**Relay logs:** MySQL automatically manages the set of relay log files, removing files when it has replayed all events and creating a new file when the current file exceeds the maximum relay log file size (or `max_binlog_size` if `max_relay_log_size` is 0). The relay logs are stored in the same format as the binary logs; you can view them with `mysqlbinlog`. The slave maintains an index file to keep track of relay log files.

The relay log files are named `<host_name>-relay-bin.<nnnnnn>` by default, and the index file is named `<host_name>-relay-bin.index`. To make the server configuration immune to potential future host name changes, change these host name prefixes by setting the options `--relay-log` and `--relay-log-index`.

**Slave status logs:** The slave stores information about how to connect to the master and the most recently replicated log coordinates of the master's binary log and the slave's relay log.

There are two such logs:

- **Master information:** This log contains information about the master server, including information such as the host name and port, credentials used to connect, and the most recently downloaded log coordinates of the master's binary log.
- **Relay log information:** This log contains the most recently executed coordinates of the relay log and the number of seconds by which the slave's replicated events are behind those of the master.

## Crash-Safe Replication

- Binary logging is crash-safe:
  - MySQL logs only complete events or transactions.
  - Use `sync-binlog` to improve safety.
    - Default value is 1; safest, the operating system writes to the file after every transaction.
    - Set the value to 0; best performance but highest possibility of transaction lost when server crashes as the operating system writes to the file according to its internal rules.
    - Set the value to any number greater than 1 to write after that number of transactions.
- Store slave status logs in tables for crash-safe replication.
  - Options: `master-info-repository` and `relay-log-info-repository`
    - Possible values are `TABLE` (the default) and `FILE`.
    - `TABLE` is crash-safe.
- Recover the relay logs to protect against logs corruption.
  - Set `relay-log-recovery` option to `ON` to create new relay log file and initialize the SQL and IO threads to continue the replication after the slave server restarts.



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

## Crash-Safe Replication

As compared to previous versions of MySQL, MySQL 8.0 has set the default values to make the replication crash-safe.

The `sync-binlog` variable now defaults to 1, and both `master-info-repository` and `relay-log-info-repository` default to `TABLE`.

Also, the binary logging is enabled by default.

## Topics

- Replication overview
- Replication conflicts
- When to use replication
- Configuring replication
- Binary and Replication Logs
- Types of Replication
- Cloning MySQL Data

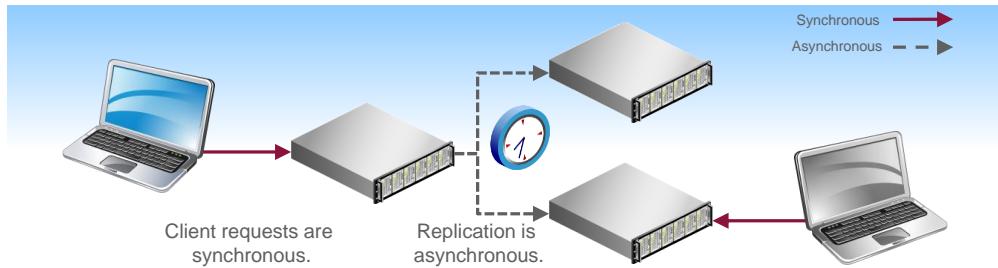


ORACLE®

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

## Asynchronous Replication

- The slave requests the binary log and applies its contents.
  - The slave typically lags behind the master.
- The master does not care when the slave applies the log.
  - The master continues operating without waiting for the slave.



ORACLE

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

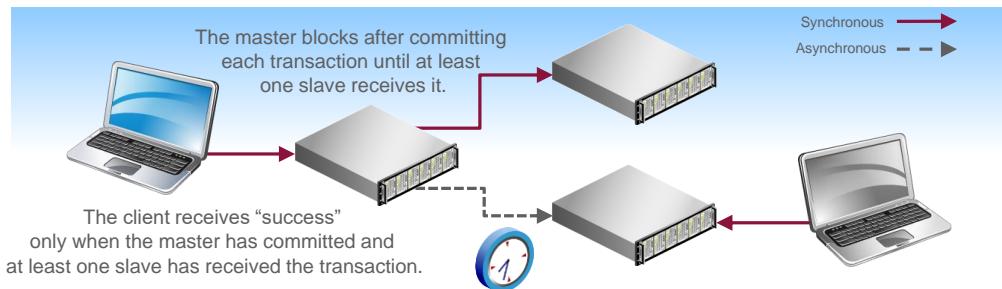
During MySQL replication in its default configuration, the master server accepts change events from clients, committing those events and writing them to the binary log. In a separate thread, the master streams the binary log to connected slaves. Because the master commits changes without waiting for a response from any slave, this is called *asynchronous* replication.

Most importantly, this means that slaves have not yet applied transactions at the time the master reports success to the application. Usually, this is not a problem. However, if the master crashes with data loss after committing a transaction but before the transaction replicates to any slave, the transaction is lost even though the application has reported success to the user.

If the master waited for all slaves to apply its changes before committing the transaction, replication would then be called *synchronous*. Although MySQL replication is not synchronous, MySQL NDB Cluster uses synchronous replication internally to ensure data consistency throughout the cluster, and MySQL client requests are synchronous because the client waits for a server response after issuing a query to the server.

## Semisynchronous Replication

- Requires a plugin on the master and at least one slave
- Blocks each master event until at least one slave receives it
- Switches to asynchronous replication if a timeout occurs



ORACLE®

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

During semisynchronous replication, the master blocks after committing a transaction until at least one semisynchronous slave acknowledges that it too has received the transaction. This means that at least one slave has received each transaction at the time the master reports success to the application. If the master crashes with data loss after committing a transaction and the application has reported success to the user, the transaction also exists on at least one slave.

## Advantages and Disadvantages of Semisynchronous Replication

Semisynchronous replication:

- Ensures data integrity
- Results in reduced performance
  - The master waits for the slave to respond before committing the transaction.
    - The timeout period is controlled by the `rpl_semi_sync_master_timeout` variable value. The default is 10,000 ms (10 seconds).
    - If no response is received, the master still commits the transaction but reverts to asynchronous mode.
  - The extra time taken by each transaction includes:
    - The TCP/IP roundtrip to send the commit to the slave
    - The slave recording the commit in its relay log
    - The master waiting for the slave's acknowledgment of that commit
- Is suitable primarily for servers that are physically co-located, communicating over fast networks



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

## Enabling Semisynchronous Replication

- Install the following plugins on the master and on at least one slave:
  - `rpl_semi_sync_master` on the master
  - `rpl_semi_sync_slave` on one or more slaves
- Enable the following options:
  - `rpl_semi_sync_master_enabled` on the master
  - `rpl_semi_sync_slave_enabled` on slaves
- Configure the `rpl_semi_master_timeout` variable if required.
  - Specifies the length of time a master will wait for a response from a semisynchronous slave.
    - After this amount of time, the master commits the transaction and reverts to asynchronous mode.
  - The default value is 10,000 milliseconds (10 seconds).



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

## Multi-Source Replication

- Enables a replication slave to receive transactions from multiple masters simultaneously
  - It requires at least two masters and one slave.
  - The slave creates a *replication channel* for each master.
  - The slave must use TABLE-based repositories.
    - Multi-source replication is not compatible with FILE-based repositories.
- Does not attempt to detect or resolve conflicts
  - If this functionality is required, it is the responsibility of the application.
- Enables:
  - Backing up multiple servers to a single server
  - Consolidating data from multiple servers to a single server
  - Merging of table shards



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

## Avoiding Conflicts

You can avoid conflicts by not replicating the same table from multiple masters.

If you need to replicate the same table from multiple masters into a single table in the slave server, the rows from different masters must be disjoint and any updates issued by a master server must be restricted to the rows in the same server.

- You can use a multicolumn primary key with one of the column to identify the master server.
- If you use statement based logging, all UPDATE and DELETE statements must have a WHERE clause on the column that identifies the master server.

### Note

Multi-source replication is compatible with auto-positioning.

## Configuring Multi-Source Replication for a GTID-Based Master

1. Enable GTID-based transactions on the master by using `gtid_mode=ON`.
2. Create a replication user.
3. Verify that the slave is using TABLE-based replication repositories.
4. Execute `CHANGE MASTER TO...` to add a new master to a channel by using a `FOR CHANNEL <channel>` clause.
  - Example: Add a new master with host name `master1` using port 3451 to channel `master-1`.

```
CHANGE MASTER TO MASTER_HOST='master1', MASTER_PORT=3451,
MASTER_USER='rpl', MASTER_PASSWORD='pass',
MASTER_AUTO_POSITION = 1
FOR CHANNEL 'master-1';
```



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

## Configuring Multi-Source Replication for a Binary Log Based Master

1. Enable binary logging on the master by using --log-bin.
2. Create a replication user.
3. Note the current binary log file and position.
  - MASTER\_LOG\_FILE and MASTER\_LOG\_POSITION
4. Verify that the slave is using TABLE-based replication repositories.
5. Execute CHANGE MASTER TO... to add a new master to a channel by using a FOR CHANNEL <channel> clause.
  - Example: Add a new master with host name master1 using port 3451 to channel master-1.

```
CHANGE MASTER TO MASTER_HOST='master1', MASTER_PORT=3451,
MASTER_USER='rpl', MASTER_PASSWORD='pass',
MASTER_LOG_FILE='master1-bin.000003', MASTER_LOG_POS=719,
FOR CHANNEL 'master-1';
```



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

## Controlling Slaves in a Multi-Source Replication Topology

- If you have enabled multiple replication channels on a slave, you can execute START SLAVE, STOP SLAVE, or RESET SLAVE for specific channels with the FOR CHANNEL clause.
- If you do not use a FOR CHANNEL clause, the statement affects all currently configured replication channels.
- Examples:

- Start replication on channel-1:

```
START SLAVE FOR CHANNEL 'channel-1';
```

- Stop replication on channel-1:

```
STOP SLAVE FOR CHANNEL 'channel-1';
```

- Reset all configured channels:

```
RESET SLAVE;
```



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

See the lesson titled “Administering a Replication Topology” for the details of these commands.

## Topics

- Replication overview
- Replication conflicts
- When to use replication
- Configuring replication
- Binary and Replication Logs
- Types of Replication
- Cloning MySQL Data



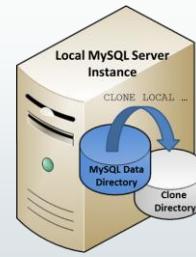
ORACLE®

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

## MySQL Clone Plugin

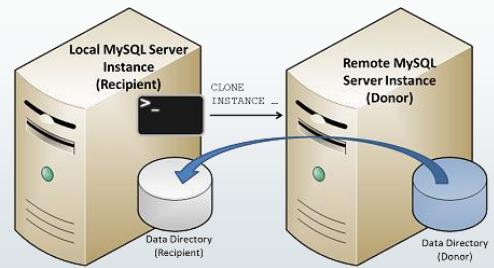
Clone the data of a MySQL server instance.

- The cloned data:
  - Contains a physical snapshot of data stored in InnoDB that includes schemas, tables, tablespaces, and data dictionary metadata
  - Can be used as the data directory to provision a MySQL server.



Two methods of cloning:

- Local cloning
  - Clone the data to a directory
- Remote cloning
  - Clone the data from a remote MySQL server instance



ORACLE®

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

The clone plugin is available as of MySQL 8.0.17.

## Installing the Clone Plugin

The **clone** plugin has to be installed on both the donor and recipient MySQL servers.

Load the **clone** plugin using one of these methods:

- Load the plugin during startup of the MySQL server using the `plugin-load` or `plugin-load-add` `startup` option:

```
[mysqld]
plugin-load-add=mysql_clone.so
```

- Add this into the option file so that it will be loaded each time the server restarts.

- Load the plugin during run time using the `INSTALL PLUGIN` statement:

```
mysql> INSTALL PLUGIN clone SONAME 'mysql_clone.so';
```

- This registers the plugin in the `mysql.plugins` system table so that it will be loaded on each subsequent server restart.



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

The plugin library file base name is `mysql_clone.so`. The file name suffix differs by platform (for example, `.so` for Unix and Unix-like systems, `.dll` for Windows).

To verify plugin installation, examine the `INFORMATION_SCHEMA.PLUGINS` table or use the `SHOW PLUGINS`.

```
mysql> SELECT PLUGIN_NAME, PLUGIN_STATUS
 FROM INFORMATION_SCHEMA.PLUGINS
 WHERE PLUGIN_NAME = 'clone';
+-----+-----+
| PLUGIN_NAME | PLUGIN_STATUS |
+-----+-----+
| clone | ACTIVE |
+-----+-----+
```

## Granting Permissions to Users

For **local** cloning:

- The user executing the cloning requires the BACKUP\_ADMIN privilege.

```
mysql> GRANT BACKUP_ADMIN ON *.* TO 'clone_user';
```

For **remote** cloning:

- The user initiating the cloning in the recipient server requires the CLONE\_ADMIN privilege for replacing the data, blocking DDL during the cloning operation, and automatically restarting the server.

```
mysql> GRANT CLONE_ADMIN ON *.* TO 'clone_user';
```

- The user account in the donor server requires the BACKUP\_ADMIN privilege for accessing and transferring data from the donor and for blocking DDL during the cloning operation. This account must accept connections from the recipient server.

```
mysql> GRANT BACKUP_ADMIN ON *.* TO 'donor_user'@'recipient_host';
```



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

## Cloning Local Data

- Clone data from the local MySQL data directory to another directory on the same host machine.

- Run the `CLONE` statement with the `LOCAL DATA DIRECTORY` option:

```
mysql> CLONE LOCAL DATA DIRECTORY = '/path/to/clone_dir';
```

- Specify an absolute directory to store the cloned data.
- The directory (`clone_dir`) must not exist, but the specified path (`/path/to`) must be an existent path.

- The cloned data directory can be used to provision a MySQL server instance:

```
shell> mysqld --datadir=/path/to/clone_dir
```

- You have to configure other required settings such as the listener port and socket file.



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

A local cloning operation does not support cloning of user-created tables or tablespaces that reside outside of the data directory. All user-created InnoDB tables and tablespaces, the InnoDB system tablespace, redo logs, and undo tablespaces are cloned to the specified directory.

## Cloning Remote Data

- Clone data from a remote MySQL server instance (the donor) to the local MySQL server instance (the recipient).
- Set the `clone_valid_donor_list` global variable to the list of donor host and port.

```
mysql> SET GLOBAL clone_valid_donor_list = 'donor_host:3306';
```

- Require the `SYSTEM_VARIABLES_ADMIN` privilege.

- Run the `CLONE` statement with the `INSTANCE FROM` option on the recipient server:

```
mysql> CLONE INSTANCE FROM 'donor_user'@'donor_host':3306
 IDENTIFIED BY 'password';
```

- Specify the donor user account, password, donor server, and port number.
- The cloning operation removes existing data in the recipient data directory, replaces it with the cloned data, and automatically restarts the server afterward.



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

There are some conditions for remote cloning:

- The donor and recipient MySQL server instances must run on the same operating system and platform.
- The recipient must have enough disk space for the cloned data.
- If the donor MySQL server instance has tablespaces that reside outside the data directory, the cloning operation must be able to write to those directories.
- Plugins that are active on the donor, including any keyring plugin, must also be active on the recipient.
- The donor and recipient must have the same MySQL server character set and collation.
- The same `innodb_page_size` and `innodb_data_file_path` settings are required on the donor and recipient.
- If cloning encrypted or page-compressed data, the donor and recipient must have the same filesystem block size.
- For page-compressed data, the recipient file system must support sparse files and hole punching for hole punching to occur on the recipient.
- A secure connection is required if you are cloning encrypted data.
- The `clone_valid_donor_list` setting on the recipient must include the host address of the donor.
- There must be no other cloning operation running.
- The `max_allowed_packet` value is at least 2MB on the donor and the recipient.

## Cloning for Replication

- Cloning can be used to provision a replication slave server.
- The cloning operation extracts the replication coordinates from the donor and transfers them to the recipient.
- For binary log replication, query the `performance_schema.clone_status` table to check the binary log position and substitute the filename and position into the `CHANGE MASTER TO` statement.

```
mysql> SELECT BINLOG_FILE, BINLOG_POSITION FROM performance_schema.clone_status;
mysql> CHANGE MASTER TO MASTER_HOST = 'donor_host', ...
 -> MASTER_LOG_FILE = 'binlog_file', MASTER_LOG_POS = binlog_position;
```

- For GTID replication, execute the `CHANGE MASTER TO` statement with `MASTER_AUTO_POSITION=1`.

```
mysql> CHANGE MASTER TO MASTER_HOST = 'master_host',..., MASTER_AUTO_POSITION=1;
```



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

## Clone Plugin Limitations

- All DDL operation is blocked during the cloning operation. Concurrent DML operations are allowed.
- Clone only the data of InnoDB tables. Other storage engine tables are cloned as empty tables with no data.
- Does not clone the server configurations
- Does not clone the binary logs
- Local cloning does not clone general tablespaces that are created with an absolute path.
- Remote cloning does not support the X Protocol port.
- Cannot connect to the donor MySQL server instance though MySQL Router



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

## Summary



In this lesson, you should have learned how to:

- Describe MySQL replication
- Explain the role of replication in high availability and scalability
- Set up a MySQL replication environment
- Design advanced replication topologies
- Clone MySQL data



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

## Practices

- 12-1: Quiz – Configuring Replication
- 12-2: Configuring Replication
- 12-3: Adding a New Slave with the Cloning Method
- 12-4: Enabling GTID and Configuring Circular Replication



ORACLE®

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

# 13

# Administering a Replication Topology



ORACLE®

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

## Objectives



After completing this lesson, you should be able to:

- Perform a controlled failover
- Explain MySQL replication threads
- Monitor MySQL replication
- Troubleshoot MySQL replication



ORACLE®

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

## Topics

- Failover
  - Replication threads
  - Monitoring replication
  - Troubleshooting replication

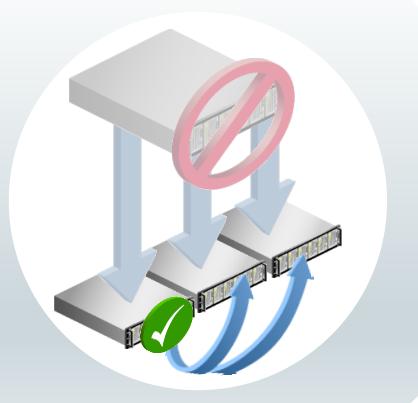


ORACLE®

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

## Failover with Log Coordinates

1. In the binary logs, find the most recent event applied to each slave.
2. Select an up-to-date slave as a new master.
3. Identify the log coordinates on the new master to match the latest applied event on each slave.
4. Issue the correct `CHANGE MASTER TO...` on each slave.



ORACLE®

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

## Potential Problems When Executing a Failover with Log Coordinates

If the slaves are not all up-to-date when you fail over, you risk having an inconsistent replication topology:

- If the new master is behind a particular slave (that is, if the slave has already applied the events at the end of the new master's log), then the slave repeats those events.
- If the new master is ahead of a particular slave (that is, if the new master's binary log contains events that the slave has not yet applied), then the slave skips those events.



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

## Avoiding Problems When Executing a Failover with Log Coordinates

- Allow the SQL thread to apply all the events in the relay log.
- Select an up-to-date slave as the new master.
- Find the log coordinates on the new master that match the most recent event on each slave.
- If some slaves are more behind than others, the log coordinates that you provide in the `CHANGE MASTER TO...` statement are different from one slave to the next.
  - You cannot simply issue a `SHOW MASTER STATUS` on the new master.
  - Instead, you must examine the binary logs to find the correct coordinates.
- In circular topologies with multiple masters accepting client updates, finding an up-to-date slave and identifying correct log coordinates can be very difficult.
  - Consider using Global Transaction Identifiers (GTIDs).



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

In circular topologies with multiple masters accepting client updates, finding an up-to-date slave and identifying correct log coordinates can be very difficult, because each slave applies operations in a different order to other slaves. To avoid this difficulty, use Global Transaction Identifiers (GTIDs).

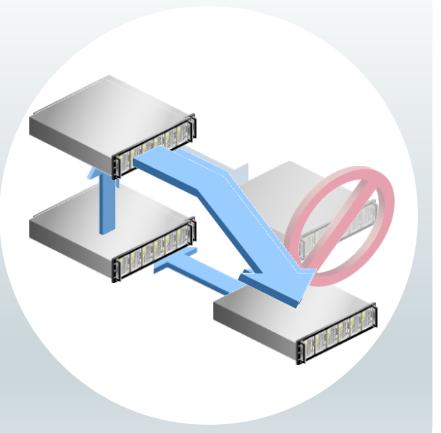
## Failover with GTIDs

When you use GTIDs, failover in a circular topology is trivial:

- On the slave of the failed master, bypass it by issuing a single `CHANGE MASTER TO` statement.
- Each server ignores or applies transactions replicated from other servers in the topology, depending on whether the transaction's GTID has already been seen or not.

Failover in a non-circular topology is similarly easy.

- Temporarily configure the new master as a slave of an up-to-date slave until it has caught up.
- On the slaves of the failed master, issue a `CHANGE MASTER TO` statement replicate from a new master.



ORACLE®

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

Although GTIDs prevent the duplication of events that originated on a single server, they do not prevent conflicting operations that originated on different servers, as described in the slide titled “Complex Topologies” in the lesson titled “Configuring a Replication Topology.” When reconnecting applications to your servers after a failover, you must be careful not to introduce such conflicts.

## Topics

- Failover
- Replication threads
- Monitoring replication
- Troubleshooting replication



ORACLE®

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

## Replication Threads

When a slave connects to a master:

- The master creates a *Binlog dump thread*
  - Reads events from the binary log and sends them to the slave I/O thread
- The slave creates two threads by default:
  - *Slave I/O thread*
    - Reads events from the master's Binlog dump thread and writes them to the slave's relay log
  - *Slave SQL (or "applier") thread*
    - Applies relay log events on single-threaded slave, or
    - Distributes relay log events between worker threads on multithreaded slave



ORACLE®

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

## The Master's Binlog Dump Thread

- The master creates one instance of the Binlog dump thread for each connected slave.
- The Binlog dump thread:
  - Displays as `Binlog Dump GTID` if the slave uses auto-positioning
    - Configure auto-positioning with `CHANGE MASTER TO MASTER_AUTO_POSITION`.
  - Sends events within the binary log to the slave as those events arrive
    - For as long as the slave is connected



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

## Single-Threaded Slaves

- Slaves are *single-threaded* by default.
  - Each slave uses a single SQL thread to process the relay log.
  - The benefit is that data within one database, when replicated by a single thread, is guaranteed to be consistent.
- Single-threaded slaves can result in *slave lag*, where the slave falls behind the master:
  - The master applies changes in parallel if it has multiple client connections, but it serializes all events in its binary log.
  - The slave executes these events sequentially in a single thread, which can become a bottleneck in high-volume environments or when slave hardware is not powerful enough.



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

## Multithreaded Slaves

- Use multithreaded slaves to reduce slave lag.
- Set the `slave_parallel_workers` variable to a value greater than zero to create that number of worker threads.
  - With multiple worker threads, the slave SQL thread does not apply events directly, but delegates responsibility to worker threads.
- Set the `slave_parallel_type` variable to specify the parallelization policy.
  - `DATABASE` (default): Transactions that update different databases are applied in parallel.
  - `LOGICAL_CLOCK`: Transactions that are part of the same binary log group commit on a master are applied in parallel on a slave.
    - The master server can configure the binary log to record commit timestamps or write sets using the `binlog_transaction_dependency_tracking` variable.
    - Set `slave_preserve_commit_order` variable to `ON` to preserve the commit order.



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

`DATABASE` policy is only appropriate if data is partitioned into multiple databases that are being updated independently and concurrently on the master. There must be no cross-database constraints, as such constraints may be violated on the slave.

`LOGICAL_CLOCK` policy uses either the commit timestamps or write sets to determine which transactions can be applied in parallel. The dependency information is generated by the master server. Write sets can provide a higher level of parallelism compared to commit timestamps. With `slave_preserve_commit_order` enabled, the executing thread waits until all previous transactions are committed before committing. With this mode, a multithreaded slave never enters a state that the master was not in.

## Controlling Slave Threads

- Start or stop both the I/O and SQL threads on a slave:

```
START SLAVE;
STOP SLAVE;
```

- Control threads individually:

```
START SLAVE IO_THREAD;
STOP SLAVE SQL_THREAD;
```

- You cannot control individual worker threads in a multithreaded slave, because these are controlled by the SQL thread.

- Start threads until a specified condition:

```
START SLAVE UNTIL SQL_AFTER_MTS_GAPS;
START SLAVE SQL_THREAD UNTIL SQL_BEFORE_GTIDS = '0ed18583-47fd-11e2-
92f3-0019b944b7f7:338';
```

- Disconnect the slave from the master:

```
RESET SLAVE [ALL]
```



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

## Resetting the Slave

Disconnect the slave from the master for a clean start by issuing the **RESET SLAVE** command:

- Clears `master.info` and `relay.log` repositories
- Deletes all the relay logs
- Starts a new relay log file
- Resets any `MASTER_DELAY` specified in the `CHANGE MASTER TO` statement to zero
- Retains connection parameters so that you can restart the slave without executing `CHANGE MASTER TO`.
  - Issue **RESET SLAVE ALL** to reset the connection parameters.
- Does not change the values of `gtid_executed`, `gtid_purged`, or the `mysql.gtid_executed` table.



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

## Quiz



Which of the following statements are true? (Select all that apply.)

- a. A single-threaded replication slave always has at least two threads.
- b. The number of active worker threads in a multithreaded slave might be affected by the number of databases in the replication network.
- c. The slave I/O thread is responsible for processing the relay log.
- d. The master sends binary log events to the slave using the Binlog dump thread.



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

**Answer: a, b, d**

## Topics

- Failover
- Replication threads
- Monitoring replication
- Troubleshooting replication



ORACLE®

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

## Monitoring Replication

```
mysql> SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State: Queueing master event to the relay log
...
Master_Log_File: mysql-bin.000005
Read_Master_Log_Pos: 79
 Relay_Log_File: slave-relay-bin.000005
 Relay_Log_Pos: 548
Relay_Master_Log_File: mysql-bin.000004
 Slave_IO_Running: Yes
 Slave_SQL_Running: Yes
...
Exec_Master_Log_Pos: 3769
...
Seconds_Behind_Master: 8
```



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

See <https://dev.mysql.com/doc/refman/en/show-slave-status.html> for the description of all the fields returned by SHOW SLAVE STATUS statement.

## Slave Thread Status

- The **slave\_IO\_Running** and **Slave\_SQL\_Running** columns display the status of the slave's I/O and SQL thread, respectively.
- Possible values for each are:
  - Yes: The thread is currently running.
  - No: The thread is not running.
  - Connecting: The thread is running, but not yet connected to the master.



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

## Master Log Coordinates

The `Master_Log_File` and `Read_Master_Log_Pos` columns identify the coordinates of the most recent event in the master's binary log that the I/O thread has transferred.

- Compare the master log coordinates with the coordinates shown when you execute `SHOW MASTER STATUS` on the master.
- If the values of `Master_Log_File` and `Read_Master_Log_Pos` are far behind those on the master, it indicates latency in the network transfer of events between the master and slave.



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

## Relay Log Coordinates

- The `Relay_Log_File` and `Relay_Log_Pos` columns identify the coordinates of the most recent event in the slave's relay log that the SQL thread has executed.
- The `Relay_Master_Log_File` and `Exec_Master_Log_Pos` columns correspond to coordinates in the master's binary log.
  - If the columns' output is far behind the `Master_Log_File` and `Read_Master_Log_Pos` columns that represent the coordinates of the I/O thread:
    - There is latency in the SQL thread rather than the I/O thread
    - The log events are being copied faster than they are being executed
- The `Relay_Log_Space` specifies the total combined size of all relay log files.
- The `Seconds_Behind_Master` column stores the number of seconds between the most recent relay log event and the current server time and helps to identify slave lag.
- The `SQL_Delay` specifies the number of seconds that the slave must lag the master as configured in the `CHANGE MASTER TO` command.



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

- **Exec\_Master\_Log\_Pos:** On a multithreaded slave, `Exec_Master_Log_Pos` contains the position of the last point before any uncommitted transactions. This is not always the same as the most recent log position in the relay log, because a multithreaded slave might execute transactions on different databases in a different order from that represented in the binary log.
- **Seconds\_Behind\_Master:** This column provides the number of seconds between the time stamp (on the master) of the most recent event in the relay log executed by the SQL thread and the real time of the slave machine. A delay of this type normally occurs when the master processes a large volume of events in parallel that the slave must process serially, or when the slave's hardware is not capable of handling the same volume of events that the master can handle during periods of high traffic. If the slave is not connected to the master, this column is `NULL`. Large transaction or long execution time such as updates to large table without primary key can cause this value to increase.

**Note:** This column does not show latency in the I/O thread or in the network transfer of events from the master.

## Replication Slave I/O Thread States

The most commonly seen I/O thread states (in the `state` column of the `SHOW PROCESSLIST` output) are:

- **Connecting to master**
  - The thread is attempting to connect to the master.
- **Waiting for master to send event**
  - The slave thread has connected and is waiting for binary log events.
  - Can last a long time if the master is idle
  - Times out after `slave_read_timeout` seconds and attempts to reconnect
- **Queuing master event to the relay log**
  - The thread has read an event and is copying it to the relay log for processing by the SQL thread.



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

## Replication Slave I/O Thread States

- **Waiting to reconnect after a failed binlog dump request**
  - The binary log dump request failed due to disconnection.
  - The thread enters this state while it sleeps and attempts to reconnect periodically.
  - Specify the interval between retries by using the `MASTER_CONNECT_RETRY` option of `CHANGE MASTER TO`.
- **Reconnecting after a failed binlog dump request**
  - The thread is trying to reconnect to the master.
- **Waiting to reconnect after a failed master event read**
  - A disconnect occurred during reading. The thread sleeps for a specified number of seconds before attempting to reconnect.
    - The default is 60 seconds.
  - Specify the number of seconds by using the `MASTER_CONNECT_RETRY` option of `CHANGE MASTER TO`.



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

## Replication Slave I/O Thread States

- **Reconnecting after a failed master event read**
  - The thread is trying to reconnect to the master.
  - When the thread reconnects, the state becomes Waiting for master to send event.
- **Waiting for the slave SQL thread to free enough relay log space**
  - The combined size of the relay logs exceeds the value of `relay_log_space_limit`.
    - Only if non-zero. A zero value means there is no limit imposed on the size of the relay logs.
  - The I/O thread is waiting until the SQL thread frees enough space in the relay logs by processing and then deleting their contents.



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

## Replication Slave SQL Thread States

The most commonly seen SQL thread states are:

- **Waiting for the next event in relay log**
  - The initial state before Reading event from the relay log
- **Reading event from the relay log**
  - The thread has read an event from the relay log that can be processed.
- **Making temp file (append) before replaying LOAD DATA INFILE**
  - The thread is executing a LOAD DATA statement and is appending the data to a temporary file containing the data from which the slave will read rows.
- **Slave has read all relay log; waiting for more updates**
  - The thread has processed all events in the relay log files. It is now waiting for the I/O thread to write new events to the relay log.



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

## Replication Slave SQL Thread States

- **Waiting until ... seconds after master executed event**
  - The SQL thread has read an event but is waiting for the slave delay to lapse.
  - Set the delay with the `MASTER_DELAY` option of `CHANGE MASTER TO`.
- **Waiting for an event from Coordinator**
  - It appears only in worker threads on multithreaded slaves.
  - A worker is waiting for the coordinating SQL thread to assign a job to the worker queue.



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

## Monitoring Replication by Using Performance Schema

The Performance Schema contains `replication_*` tables that improve on the information available via `SHOW SLAVE STATUS`, enabling you to:

- Exercise more control over what information you want to display
- Retrieve better diagnostic information
  - `SHOW SLAVE STATUS` reports only the most recent coordinator and worker thread errors.
  - Retrieve details of the last transaction handled by each worker thread.
- Persist diagnostic information in tables or views
- Access replication metrics programmatically



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

## Replication Tables in Performance Schema

| Table Name                                             | Contains                                                                                       |
|--------------------------------------------------------|------------------------------------------------------------------------------------------------|
| <code>replication_connection_configuration</code>      | Configuration parameters for connecting to the master                                          |
| <code>replication_connection_status</code>             | Status of the current connection to the master                                                 |
| <code>replication_applier_configuration</code>         | Configuration parameters for the transaction applier on the slave                              |
| <code>replication_applier_status</code>                | Current status of the transaction applier on the slave                                         |
| <code>replication_applier_status_by_coordinator</code> | Status of the coordinator thread in a multithreaded slave                                      |
| <code>replication_applier_status_by_worker</code>      | Status of applier thread in a single-threaded slave or worker threads in a multithreaded slave |
| <code>replication_applier_filters</code>               | Information about the replication filters configured on specific replication channels          |
| <code>replication_applier_global_filters</code>        | Information about global replication filters (all channels)                                    |
| <code>replication_group_members</code>                 | Network and status information for group members                                               |
| <code>replication_group_member_stats</code>            | Statistical information about group members and the transactions they participate in           |

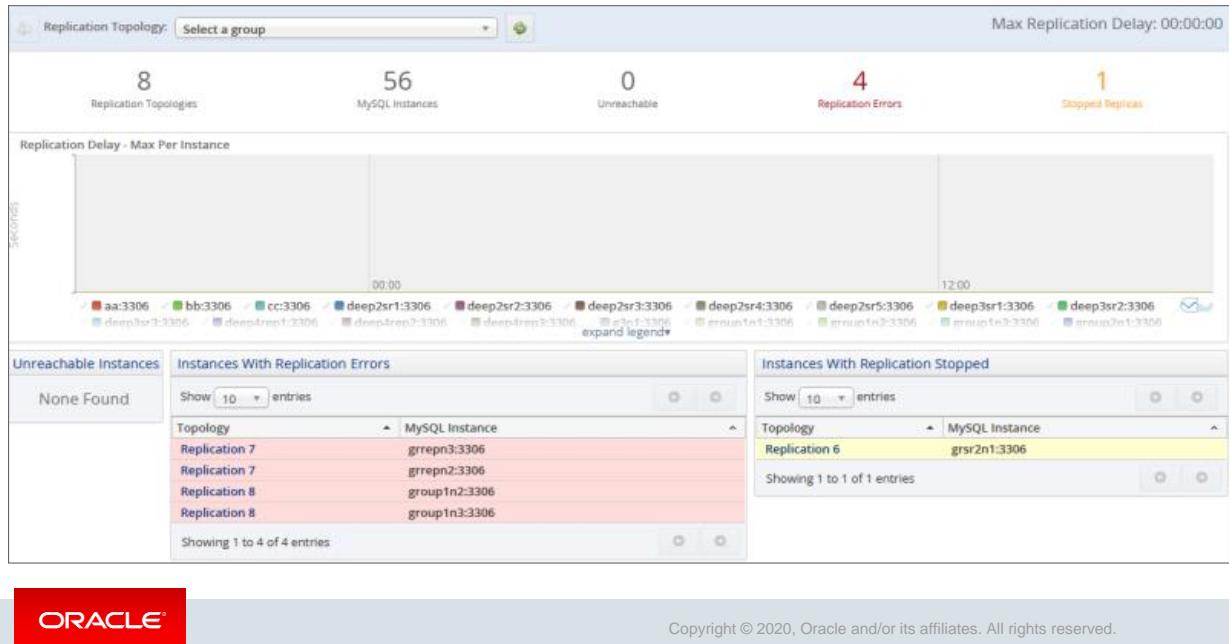


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

These tables can give you more information as compared to the `SHOW SLAVE STATUS` statement. You can check the status of individual worker threads in a multithreaded slave that is not available with the `SHOW SLAVE STATUS` statement.

The contents of the Performance Schema `log_status` table can be useful for replication. It contains the status of the `gtid_executed`, local binary logs, and replication relay logs information.

## MySQL Enterprise Monitor Replication Dashboard



The Replication Dashboard displays all information related to monitored replication groups. MySQL Enterprise Monitor supports monitoring of single-source tree hierarchy; circular replication; group replication; or complex, multi-level, multisource hierarchies.

Navigate to the Replication page by choosing Replication under Dashboards. This page summarizes the state of your replication servers and enables you to drill down to see details about any source or replica. Using this page helps you avoid running the `SHOW SLAVE STATUS` command repeatedly on multiple servers; for consistency, the Replication page uses some of the same keywords as the output from `SHOW SLAVE STATUS`.

## Topics

- Failover
- Replication threads
- Monitoring replication
- Troubleshooting replication



ORACLE®

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

## Troubleshooting MySQL Replication

- View the error log.
  - The error log can provide you with enough information to identify and correct problems in replication.
- Issue a `SHOW MASTER STATUS` statement on the master.
  - Logging is enabled if the position value is non-zero.
- Verify that both the master and slave have a unique non-zero server ID value.
  - The master and the slave must have different server IDs.
- Issue a `SHOW SLAVE STATUS` command on the slave or query the replication tables in Performance Schema.
  - `Slave_IO_Running` and `Slave_SQL_Running` display `Yes` when the slave is functioning correctly.
  - `Last_IO_Error` and `Last_SQL_Error` show the most recent error messages from the I/O and SQL threads.

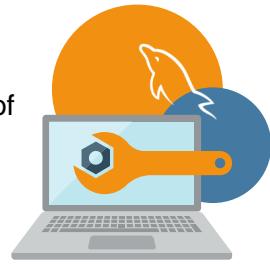


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

**Note:** If you are unable to log in to the slave by using a valid account on the master, it might be because you created the user account with `CREATE USER` or `GRANT` on the master, and it replicated to the slave via the `mysql.user` table, but the replication process did not flush privileges. Use the `flush-privileges` command with `mysqladmin` or execute `FLUSH PRIVILEGES` on the slave from another account.

## Troubleshooting MySQL Replication

- Issue a `SHOW PROCESSLIST` command on the master and slave.
  - Review the state of the Binlog dump, I/O, and SQL threads.
- For a slave that suddenly stops working, check the most recently replicated statements.
  - The SQL thread stops if an operation fails due to a constraint problem or other error.
    - The error log contains events that cause the SQL thread to stop.
  - Review known replication limitations.
    - <http://dev.mysql.com/doc/mysql/en/replication-features.html>
  - Verify that the slave data has not been modified directly (outside of replication).



ORACLE®

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

## Examining the Error Log

- The error log stores information about when replication begins and about any errors that occur during replication.
- Always check the error log first.
- The following example shows a successful start and then a subsequent failure when replicating a user that already exists on the slave:

```
date-and-time 95 [System] [MY-010562] [Rep1] Slave I/O thread for channel '':
connected to master 'repl@10.0.0.23:3306', replication started in log 'binlog.000005'
at position 1384
...
date-and-time 96 [ERROR] [MY-010584] [Rep1] Slave SQL for channel '':
Error 'Operation
CREATE USER failed for 'user'@'%' on query. Default database: ''.
Query: 'CREATE USER
'user'@'%' IDENTIFIED WITH 'caching_sha2_password' AS '....'', Error_code: MY-001396
date-and-time 96 [Warning] [MY-010584] [Rep1] Slave: Operation CREATE USER failed for
'user'@'%' Error_code: MY-001396
date-and-time 96 [ERROR] [MY-010586] [Rep1] Error running query, slave SQL thread
aborted. Fix the problem, and restart the slave SQL thread with "SLAVE START". We
stopped at log 'binlog.000006' position 235
```



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

## Examining the Error Log

The following sequence shows a failure of the I/O thread reading from the master's binary log:

```
date-and-time 1421 [System] [MY-010562] [Rep1] Slave I/O thread for channel '':
connected to master 'repl@10.0.0.23:3306', replication started in log 'binlog.000007'
at position 200
...
date-and-time 1421 [ERROR] [MY-010557] [Rep1] Error reading packet from server for
channel ''': bogus data in log event; the first event 'binlog.000007' at 200, the last
event read from './binlog.000007' at 124, the last byte read from './binlog.000007' at
219. (server_errno=1236)
date-and-time 1421 [ERROR] [MY-013114] [Rep1] Slave I/O for channel ''': Got fatal
error 1236 from master when reading data from binary log: 'bogus data in log event;
the first event 'binlog.000007' at 200, the last event read from './binlog.000007' at
124, the last byte read from './binlog.000007' at 219.', Error_code: MY-013114
```



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

## SHOW SLAVE STATUS Error Details

- **Last\_IO\_Error, Last\_SQL\_Error:**
  - The error message of the most recent error that caused, respectively, the I/O thread or SQL thread to stop
  - During normal replication, these fields are empty.
  - If an error occurs and causes a message to appear in either of these fields, the error message also appears in the error log.
- **Last\_IO\_Errno, Last\_SQL\_Errno:**
  - The error number associated with the most recent error that caused, respectively, the I/O or SQL thread to stop
  - During normal replication, these fields contain the number 0.
- **Last\_IO\_Error\_Timestamp, Last\_SQL\_Error\_Timestamp:**
  - The time stamp of the most recent error that caused, respectively, the I/O thread or SQL thread to stop
  - During normal replication, these fields are empty.



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

## Checking I/O Thread States

If the slave is running, issue `SHOW PROCESSLIST` to check the I/O thread states connecting to the master:

- Verify privileges for the user being used for replication on the master
- Verify that the host name and port are correct for the master
- Verify that networking has not been disabled on the master or the slave (with the `--skip-networking` option)
- Attempt to ping the master to verify that the slave can reach the master



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

## Monitoring Multi-Source Replication

- Execute `SHOW SLAVE STATUS FOR CHANNEL <channel name>`.
- Alternatively, query the Performance Schema `replication_*` tables, specifying the channel name.
  - The `CHANNEL_NAME` field exists in every Performance Schema replication table.

```
mysql> SELECT * FROM replication_connection_status WHERE CHANNEL_NAME='master1' \G;
***** 1. row *****
CHANNEL_NAME: master1
GROUP_NAME:
SOURCE_UUID: 046e41f8-a223-11e4-a975-0811960cc264
THREAD_ID: 24
SERVICE_STATE: ON
COUNT_RECEIVED_HEARTBEATS: 0
LAST_HEARTBEAT_TIMESTAMP: 0000-00-00 00:00:00
RECEIVED_TRANSACTION_SET: 046e41f8-a223-11e4-a975-0811960cc264:4-37
LAST_ERROR_NUMBER: 0
LAST_ERROR_MESSAGE:
LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00
```



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

The example in the slide demonstrates querying the Performance Schema `replication_connection_status` table to monitor the status of the I/O thread that handles the slave server connection to the master server on replication channel `master1`.

## Summary



In this lesson, you should have learned how to:

- Perform a controlled failover
- Explain MySQL replication threads
- Monitor MySQL replication
- Troubleshoot MySQL replication



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

## Practices

- 13-1: Monitoring Replication with MySQL Enterprise Monitor
- 13-2: Troubleshooting Replication Errors
- 13-3: Performing a Replication Failover



ORACLE®

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

# Achieving High Availability with MySQL InnoDB Cluster

14



ORACLE®

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

## Objectives



After completing this lesson, you should be able to:

- Describe MySQL InnoDB cluster and Group Replication
- List typical use cases for MySQL InnoDB cluster
- Contrast the two different modes for deployment
- Configure a MySQL InnoDB cluster
- Administer a MySQL InnoDB cluster



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

## Topics

- Overview and architecture
- Tools
- Configuring a cluster
- Administering a cluster



ORACLE®

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

## What Is MySQL InnoDB Cluster?

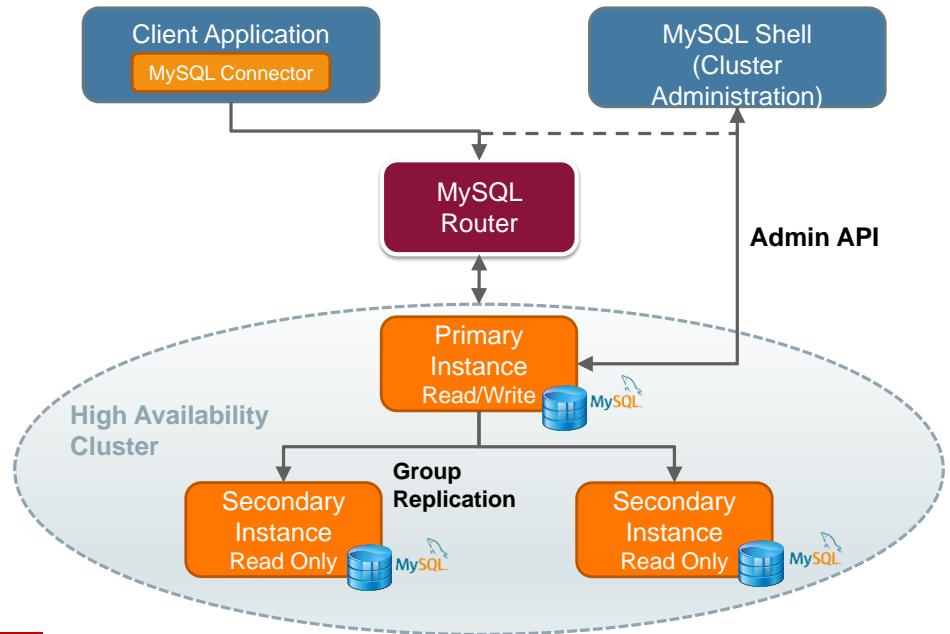
- Provides a complete and scalable high availability solution for MySQL
  - Easy to configure and administer a group of server instances in a cluster
- Uses **MySQL Group Replication** to replicate data between all servers in the group
  - The **AdminAPI** removes the need to work with group replication directly.
  - You work with the AdminAPI via **MySQL Shell** using your choice of Python or JavaScript language.
- Manages failover automatically
  - If a server in the group goes down, the cluster reconfigures itself.
  - It requires at least three servers for the group to be fault tolerant
- Enables clients to connect to the group transparently
  - Clients connect to the group via **MySQL Router** and do not have to know the details of the individual instances within the group.



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

To use MySQL InnoDB cluster, you must download and install the MySQL Shell and MySQL Router software.

## Architecture



ORACLE

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

The cluster relies on MySQL Group Replication, which is installed on each server instance in the cluster. Group Replication is a MySQL server plugin that enables you to create elastic replication topologies that can reconfigure themselves automatically if a server in the cluster goes offline. There must be at least three servers to form a group that can provide high availability. Groups can operate in a single-primary mode where only one server accepts updates at a time or multi-primary mode where all servers can accept updates, even if they are issued concurrently.

The MySQL Group Replication plugin was introduced in 5.7, but it is tricky to work with directly. MySQL InnoDB cluster introduces new components that are tightly integrated and make Group Replication much easier to set up and administer. These include MySQL Router, which sits between the application and the cluster and routes the traffic to the appropriate cluster instance. If the primary instance goes down, the cluster automatically promotes another instance to take its place, and the MySQL Router starts routing traffic to the new primary instance. All this happens without DBA intervention.

You administer the cluster by using MySQL Shell. MySQL Shell is a new interactive interface that enables you to administer MySQL via the new Admin API, using familiar JavaScript or Python syntaxes.

## MySQL Group Replication Plugin

Group Replication is a plugin for MySQL that enables a group of servers to replicate data between them and provides:

- Automatic handling of server failover
- Automatic reconfiguration of groups when members join or leave the group due to crashes, failures, or reconnects
- Fault tolerance
- Conflict resolution
- A highly available, replicated database



ORACLE®

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

## How Group Replication Works

- Servers belong to a *replication group*.
  - A replication group can contain up to nine servers.
    - At least three servers are required to be fault tolerant.
  - Group Replication uses global transaction identifiers (GTIDs).
  - The group is defined by a UUID.
- Group membership is managed automatically.
  - A server that joins or rejoins the group will automatically synchronize with the others.
  - Servers can leave and join the group at any time.
- Replication groups operate in one of two modes:
  - **Single-primary**: One server in the group accepts updates.
  - **Multi-primary**: All servers in the group accept updates.
- Changes are replicated to all members of the group.



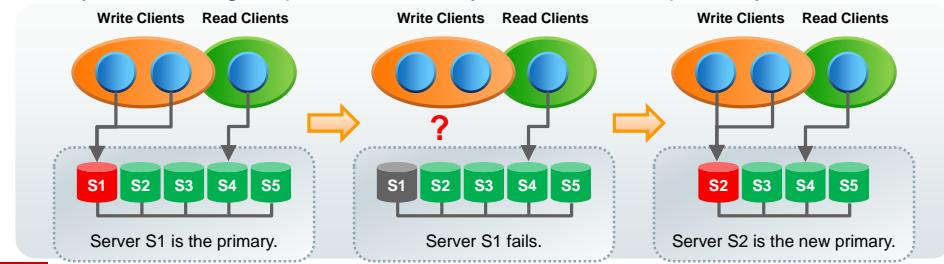
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Member servers are configured to belong to a *replication group*. Clients write to one or more members, depending on the Group Replication mode. Changes are replicated to all members of the replication group.

Groups can operate in *single-primary mode* with automatic primary election, where only one member accepts updates. Alternatively, groups can be deployed in *multi-primary mode*, where all members accept updates, even if they are issued concurrently.

## Single-Primary Mode

- One group member accepts writes (the *primary*).
- The remaining group members act as read-only hot standbys (*secondaries*).
- This is the default mode, and it works best in most scenarios.
  - Applications and developers interact only with a single server for updates.
  - This avoids some of the limitations of multi-primary mode.
  - Applications can connect to any secondary nodes to read data.
- If the primary fails, the group automatically elects a new primary:



ORACLE®

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

## Multi-Primary Mode

- All servers can receive updates, even when executed concurrently.
- However, it does not scale writes because all servers still need to write all the updates.
- If one server fails, clients can connect to any other server in the replication group.
- Has more restrictions as compared to single-primary mode



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

## Conflict Resolution

- Each server in the group executes transactions independently, but all servers must agree on the decision.
  - Read-write transactions commit when they are received by the group.
    - For the transaction to be applied, the majority of members must agree (“certify”) with the order of the transaction within the global sequence of transactions.
  - Read-only transactions do not require group approval and commit immediately.
- If two concurrent transactions affect a single row, there is a conflict.
  - The first transaction that commits “wins.” The others roll back.
- If a network partition results in a “split brain” situation where group members are unable to agree, then the system does not achieve consensus and requires manual intervention.



ORACLE®

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

The “first transaction to commit wins” rule also applies to single-primary mode.

## Consensus and Quorum

- The group needs to achieve consensus whenever a change that needs to be replicated happens, such as regular transactions and group membership changes.
- Consensus or quorum requires a majority (more than half) of group members to agree on a given decision.
- *Quorum* may be *lost* when there are multiple involuntary failures, causing a majority of servers to be removed abruptly from the group.
  - The remaining servers cannot determine if the other servers have crashed or whether a network partition has isolated them.
  - The remaining servers cannot be reconfigured automatically. Manual intervention is required.
- Any servers that exit the group voluntarily can inform other members in the group to reconfigure themselves. Therefore, the quorum can be maintained.



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

For example in a group of five servers, if three of them become silent at once, the majority is compromised and thus no quorum can be achieved. In fact, the remaining two are not able to tell if the other three servers have crashed or whether a network partition has isolated these two alone, and therefore the group cannot be reconfigured automatically. Refer to the slide titled “Restoring Quorum Loss” for the process to restore the quorum of a cluster.

However, in the above scenario of five servers where three leave at once, if the three leaving servers warn the group that they are leaving, one by one, then the membership is able to adjust itself from five to two, and at the same time, securing quorum while that happens.

## Use Cases

- Elastic replication:
  - Environments where the number of servers involved in the replication infrastructure is very fluid.
- Highly available shards:
  - Sharding is a popular approach to write scale-out.
  - Each shard can map to a replication group.
- As an alternative to standard master-slave replication:
  - Using single-primary mode, the following operations are automatic:
    - Assignment of primary/secondary roles
    - Election of new primary when current primary fails
    - Setup of read/write modes on primaries and secondaries
    - Global consistent view of which server is the primary



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

# Group Replication: Requirements and Limitations

## Required

- For communication:
  - Local network with servers in close proximity to reduce latency
- For conflict detection:
  - InnoDB storage engine
  - Primary key on every table
  - GTIDs are enabled
- For replication:
  - Binary logging and slave updates enabled
  - Binary log `ROW` format
  - Metadata must be stored in `TABLE` format
  - `transaction_write_set_extraction=XXHASH64`

## Forbidden

- General:
  - Binary log events checksum
  - Table locks and named locks
  - Replication filters
- In multi-primary mode:
  - `SERIALIZABLE` isolation level
  - Cascading foreign keys
  - Concurrent DDL and DML statements that access the same object on different members



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

You must ensure that the `transaction_write_set_extraction` variable is set to `XXHASH64`, so that while collecting rows to log them to the binary log, the server collects the write set as well. The write set is based on the primary keys of each row and is a simplified and compact view of a tag that uniquely identifies the row that was changed. Group replication uses this tag to detect conflicts.

When running in multi-primary mode, you cannot execute data definition language (DDL) and data manipulation language (DML) statements on the same object on different members, because it may result in undetectable conflicts.

## Quiz



Which of the following statements are correct about Group Replication in single-primary mode? (Select all that apply.)

- a. All servers accept writes to improve scalability.
- b. One server accepts writes, the others are read only.
- c. There must be at least two instances, one for writes and one for reads.
- d. A "split brain" situation always results when the secondaries lose connection to the primary.



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

**Answer: b**

## Topics

- Overview and architecture
- Tools
  - Configuring a cluster
  - Administering a cluster



ORACLE®

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

## MySQL Shell (mysqlsh)

- Is an advanced client and code editor for MySQL
- Provides scripting capabilities by using JavaScript, Python, or SQL commands
  - You can enter commands interactively or execute them in batches.
  - You can switch between languages by entering \js, \py, or \sql, respectively.
- Enables access to MySQL features via APIs
  - **XDevAPI:** Communicate with a MySQL server running the X Plugin to work with both relational and document data in the MySQL Document Store.
  - **AdminAPI:** Configure and administer a MySQL InnoDB cluster.
- Supports output in tab delimited, table, and JSON (JavaScript Object Notation) formats
- Interacts with a MySQL server via a global Session object
  - If using Python and JavaScript, you create the Session object by calling the `getSession()` method on the `mysqlx` module.
  - If using SQL, the Session object is created when the client connects.



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

The examples in this course use JavaScript (the default scripting language for MySQL Shell).

## Using MySQL Shell to Execute a Script

Examples:

- Loading JavaScript code from a file for batch processing:

```
$ mysqlsh --file script.js
```

- Redirecting a JavaScript file to standard input for execution:

```
$ mysqlsh < script.js
```

- Redirecting SQL to standard input for execution:

```
$ echo "show databases;" | mysqlsh --sql --uri root@127.0.0.1:3306
```

- Making a batch file executable (Linux only):

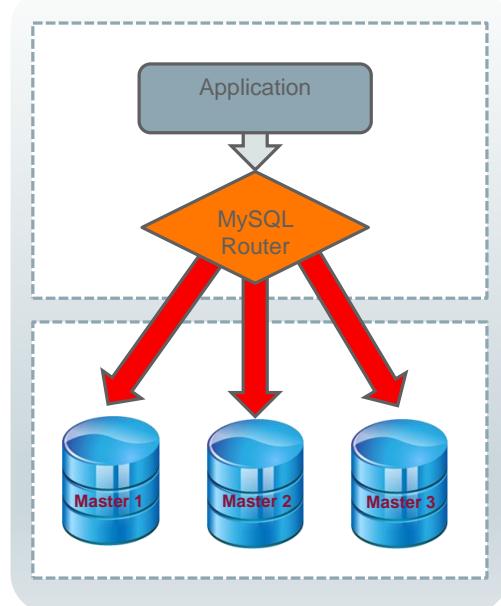
```
#!/usr/local/mysql-shell/bin/mysqlsh --file
print("Hello World\n");
```



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

## MySQL Router (`mysqlrouter`)

- Acts as middleware that provides transparent routing between client applications and back-end MySQL servers
- Manages failover by automatically routing connections
  - No custom code required; uses a load balancing policy
- Provides load balancing
  - Distributes database connections across a pool of servers for performance and scalability
- Implements a pluggable architecture
  - Developers can extend the product and create plugins for custom use cases.



ORACLE®

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

## Topics

- Overview and architecture
- Tools
- **Configuring a cluster**
- Administering a cluster



ORACLE®

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

## Deployment Scenarios

You can configure a MySQL InnoDB cluster using one of two methods:

- **Sandbox deployment:** Enables you to test MySQL InnoDB cluster locally, prior to deployment on production servers
- **Production deployment:** Enables you to deploy the individual instances that make up a cluster on multiple host machines in a network



ORACLE®

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

## Deploying Sandbox Instances and Creating the Cluster

1. Deploy three sandbox instances on ports 3310, 3320, and 3330:

```
mysql-js> dba.deploySandboxInstance(3310)
mysql-js> dba.deploySandboxInstance(3320)
mysql-js> dba.deploySandboxInstance(3330)
```

2. Connect to the seed instance (where the data resides) and create the cluster:

```
mysql-js> \connect root@localhost:3310
mysql-js> var cluster = dba.createCluster('mycluster')
```

3. Add instances:

```
mysql-js> cluster.addInstance(3320)
mysql-js> cluster.addInstance(3330)
```



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

The `dba` shell global variable represents the AdminAPI. You call its `deploySandboxInstances` method, passing the required TCP port for the instance as a parameter.

You then connect to the instance by using the `shell.connect('user@localhost:port')` method or its shorthand equivalent `\connect user@server:port`, as demonstrated in the slide example. This example uses a URI string for the connection details, but you can also use a dictionary.

When you create a cluster using `dba.createCluster()`, the operation returns a `cluster` object which you can assign to a variable. You use this object to work with the cluster, for example, to add instances or check the cluster's status. If you want to retrieve a cluster again at a later date, for example, after restarting MySQL Shell, use `dba.getCluster(clustername, options)`. Set the `connectToPrimary` option to query for and connect to a primary instance. For example:

```
cluster1 = dba.getCluster('mycluster', {connectToPrimary:false})
```

MySQL InnoDB cluster relies on `SET PERSIST` to persist instance configuration settings automatically. This is only available on instances running version 8.0.11 of MySQL Server or later with `persisted_globals_load=ON` (the default setting). On local instances that support automatic persistence, the configuration settings are stored in the instance's `mysqld-auto.cnf` file. If a local instance does not support automatic persistence, the AdminAPI attempts to write these changes to the instance's option file.

## Production Deployment

1. Perform the following steps for each machine that will be part of the cluster.
  - Ensure that the machine's name is resolvable by other machines in the cluster by one of the following methods:
    - Map the IP addresses of all the other machines to a host name.
    - Set up a DNS service.
    - Configure the `--report_host` variable in the instance MySQL configuration file.
2. Verify that each instance is correctly configured for MySQL InnoDB cluster by executing `dba.checkInstanceConfiguration(connection details)`.
  - The output of the method call lists any required changes. You can accept these changes by executing `dba.configureInstance(connection details)`.
3. Create the cluster by executing `dba.createCluster(cluster name)`.
4. Add instances by executing `cluster.addInstance(connection details)`.



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

If a remote instance is not running version 8.0.11 of MySQL Server or later with `persisted_globals_load=ON` (the default setting), the AdminAPI commands cannot automatically write to the instance's option file to persist any configuration options. It can read configuration options from the file, but it cannot change them. If you need to make changes to the remote instance's option file to configure it for MySQL InnoDB cluster usage, you must connect to the server using a tool such as SSH and run MySQL Shell directly on the instance to configure it locally by using `dba.configureLocalInstance()`.

## Distributed Recovery

- Is the process to synchronize the data of a member that joins or rejoins the cluster
- Chooses an existing member as the donor to update the joining member
- Is available in two methods:
  - Clone recovery
    - Uses the clone plugin to take a snapshot of the donor and restore to the joining member
    - Removes all existing data in the joining member
    - Available as of MySQL version 8.0.17
  - Incremental recovery
    - Replicates from a donor's binary logs and applies the transactions on the joining member
    - Fails if the required binary logs have been purged from all the cluster members



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

By default, the cluster automatically chooses the most suitable method, but you can optionally configure this behavior. You can use the `recoveryMethod` option in the `cluster.addInstance()` method to select a recovery method. The valid values are 'clone' and 'incremental'.

From version 8.0.17, by default when a new cluster is created using `dba.createCluster()` on an instance where the MySQL Clone plugin is available, the Clone plugin is automatically installed, and the cluster is configured to support cloning. The InnoDB cluster recovery accounts are created with the required `BACKUP_ADMIN` privilege to support cloning.

## Connecting Clients to the Cluster

1. Install MySQL Router on the same host as the application.
2. Bootstrap MySQL Router with the metadata server on the seed instance:

```
$ mysqlrouter --bootstrap user@hostname:port --directory=directory_path
```

3. Start MySQL Router:

```
$ directory_path/start.sh
```

- This creates read/write and read-only TCP ports.

4. Connect the client to the appropriate TCP port.
  - For example, using MySQL Shell to connect to the read/write port 6446 of mysqlrouter as the `root` user

```
$./bin/mysqlsh --uri root@localhost:6446
```

5. To verify which machine you are connected to, you can query the `port` and `hostname` server variables.



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

The `--directory` option configures MySQL Router to run from a self-contained directory. This enables you to deploy multiple instances of the router in the same host without root privileges.

## Quiz



How do you persist a remote instance's configuration settings for MySQL InnoDB cluster when the version of MySQL Server is prior to 8.0.11?

- a. Manually configure the remote instance's /etc/my.cnf file.
- b. Pass persist='true' as an option to cluster.createInstance().
- c. Ensure that the remote instance has persisted\_globals\_load set to ON.
- d. Connect to the remote instance and use MySQL Shell on the instance to execute dba.configureLocalInstance().



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

**Answer: d**

## Topics

- Overview and architecture
- Tools
- Configuring a cluster
- Administering a cluster



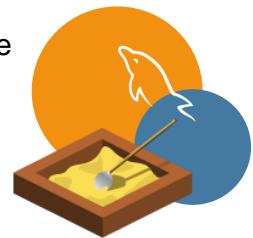
ORACLE®

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

## Managing Sandbox Instances

Manage sandbox instances with the following functions:

- `dba.deploySandboxInstance (port)` : Create a new instance.
- `dba.startSandboxInstance (port)` : Start a sandbox instance.
- `dba.stopSandboxInstance (port)` : Stop a running instance gracefully, unlike `dba.killSandboxInstance ()`.
- `dba.killSandboxInstance (port)` : Stop a running instance immediately. Useful for simulating unexpected halts.
- `dba.deleteSandboxInstance (port)` : Remove a sandbox instance from your filesystem.



ORACLE®

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

### Note

Each of these functions also accepts an optional options object as a second parameter, which affects the result of the operation.

## Checking the Status of a Cluster

```
mysql> var cluster = dba.getCluster("mycluster")
mysql> cluster.status()
{
 "clusterName": "mycluster",
 "defaultReplicaSet": {
 "name": "default",
 "primary": "localhost:3320",
 "ssl": "REQUIRED",
 "status": "OK",
 "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
 "topology": {
 "localhost:3310": {
 "address": "localhost:3310",
 "mode": "R/O",
 "readReplicas": {},
 "role": "HA",
 "status": "ONLINE"
 },
 "localhost:3320": {
 ...
 },
 "localhost:3330": {
 ...
 }
 ...
 }
 }
}
```



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

### Note

The way in which you connect to the cluster affects the output of `cluster.status()`. If you connect via a read-only connection, you will only be able to see status information that relates to the secondary instances. To see status information for the primary instance(s), use a read/write connection.

## Viewing the Structure of a Cluster

```
mysql> cluster.describe();
{
 "clusterName": "mycluster",
 "adminType": "local",
 "defaultReplicaSet": {
 "name": "default",
 "instances": [
 {
 "name": "localhost:3310",
 "host": "localhost:3310",
 "role": "HA"
 },
 {
 "name": "localhost:3320",
 "host": "localhost:3320",
 "role": "HA"
 },
 {
 "name": "localhost:3330",
 "host": "localhost:3330",
 "role": "HA"
 }
]
 }
}
```



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

The output of the `cluster.describe()` function shows the structure of the InnoDB cluster including all of its configuration information.

## Checking the State of an Instance

- Execute `cluster.checkInstanceState(instance)` to verify the instance GTID state in relation to the cluster.
  - Analyzes the instance executed GTIDs with the executed/purged GTIDs on the cluster to determine if the instance is valid for the cluster.
- The output of this method is one of the following:
  - `OK new`: The instance has not executed any GTID transactions; therefore, it cannot conflict with the GTIDs executed by the cluster.
  - `OK recoverable`: The instance has executed GTIDs, which do not conflict with the executed GTIDs of the cluster seed instances.
  - `ERROR diverged`: The instance has executed GTIDs, which diverge with the executed GTIDs of the cluster seed instances.
  - `ERROR lost_transactions`: The instance has more executed GTIDs than the executed GTIDs of the cluster seed instances.



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

## Updating a Cluster Metadata

- If an instance's configuration is changed by any means other than via the AdminAPI, you must rescan the cluster to update the InnoDB cluster metadata.
  - For example, you manually added a new instance to the Group Replication group.
- Execute `cluster.rescan()`
  - Displays any discovered instances and asks you if you want to add them to your cluster metadata



ORACLE®

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

The InnoDB cluster metadata is stored in a schema named `mysql_innodb_cluster_metadata`.

## Removing Instances from the Cluster

Remove an instance from the cluster by calling the `cluster.removeInstance()` method:

```
mysql-js> cluster.removeInstance('user@hostname:port')
```



ORACLE®

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

## Rejoining an Instance to the Cluster

- An instance that restarted will try to rejoin the cluster automatically by default.
- An instance that lost connection with other members of the cluster for more than five seconds becomes a suspected member and will be expelled after the expel timeout (`expelTimeout` option).
- An expelled instance can be configured to attempt to rejoin the cluster automatically.
  - If the auto-rejoin fails, it waits for five minutes and retries.
  - The instance switches to the configured exit action state (`exitStateAction` option) after the configured maximum number of auto-rejoin retry (`autoRejoinTries` option) fails.
- If an instance that has left the cluster (for example, due to a lost connection) does not or fails to rejoin it automatically, execute `cluster.rejoinInstance(instance)`.



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

## Restoring Quorum Loss

If an instance fails, then the cluster can lose its quorum, which is the ability to elect a new primary. Re-establish quorum with `cluster.forceQuorumUsingPartitionOf()`.

```
mysql-js> var cluster = dba.getCluster("mycluster")
// The cluster lost its quorum and its status shows "status": "NO_QUORUM"

mysql-js> cluster.forceQuorumUsingPartitionOf("localhost:3310")

Restoring replicaset 'default' from loss of quorum, by using the partition composed of
[localhost:3310]

Please provide the password for 'root@localhost:3310': *****
Restoring the InnoDB cluster ...

The InnoDB cluster was successfully restored using the partition from the instance
'root@localhost:3310'.

WARNING: To avoid a split-brain scenario, ensure that all other members of the replicaset
are removed or joined back to the group that was restored.
```



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

## Recovering the Cluster from a Major Outage

If a cluster comes to a complete standstill, perform the following steps:

1. Restart the cluster instances.
  - For example, in a sandbox deployment:

```
mysql-js> dba.startSandboxInstance(3310)
mysql-js> dba.startSandboxInstance(3320)
mysql-js> dba.startSandboxInstance(3330)
```
2. Connect to one instance and run MySQL Shell.
3. From MySQL Shell, connect to a cluster and execute `dba.rebootClusterFromCompleteOutage()`:

```
mysql-js> \connect 'root@localhost:3310'
mysql-js> var cluster = dba.rebootClusterFromCompleteOutage()
```



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

The `dba.rebootClusterFromCompleteOutage()` function reboots a cluster from complete outage. It picks the instance the MySQL Shell is connected to as new seed instance and recovers the cluster. The seed instance should have the most up-to-date data. Optionally, it also updates the cluster configuration based on user-provided options.

If this process fails, and the cluster metadata has become badly corrupted, you might need to drop the metadata and create the cluster again from scratch. You can drop the cluster metadata using `dba.dropMetadataSchema()`.

## Dissolving a Cluster

To completely dissolve a cluster:

1. Connect to a read/write instance
2. Execute `cluster.dissolve()`
  - Removes all cluster metadata and configuration and disables group replication
  - Data that was replicated between the instances remains intact
  - Nullifies any variables that were assigned to the `Cluster` object

```
mysql-js> \connect 'root@localhost:3310'
mysql-js> var cluster = dba.getCluster("mycluster")
mysql-js> cluster.dissolve()
The cluster was successfully dissolved.
Replication was disabled but user data was left intact.
```



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

## Disabling super\_read\_only

Whenever Group Replication stops, the `super_read_only` variable is set to `ON` to ensure no writes are made to the instance. When you try to use such an instance with the following AdminAPI commands, you are given the choice to set `super_read_only=OFF` on the instance:

- `dba.configureInstance()`
- `dba.configureLocalInstance()`
- `dba.createCluster()`
- `dba.rebootClusterFromCompleteOutage()`
- `dba.dropMetadataSchema()`

You can also use set the `clearReadOnly` option to `true` in non-interactive execution. For example:

- `dba.configureInstance(instance, {clearReadOnly: true})`



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

### Example of an interactive session:

```
mysql-js> var myCluster = dba.dropMetadataSchema()
Are you sure you want to remove the Metadata? [y/N]: y
The MySQL instance at 'localhost:3310' currently has the super_read_only
system variable set to protect it from inadvertent updates from
applications. You must first unset it to be able to perform any changes
to this instance.
For more information, see:
https://dev.mysql.com/doc/refman/en/server-system-variables.html#sysvar_super_read_only
```

*Do you want to disable super\_read\_only and continue? [y/N]: y*

Metadata Schema successfully removed.

## Customizing a MySQL InnoDB Cluster

When you execute `dba.createCluster(String name, Dictionary options)` to set up an InnoDB cluster, you can provide a list of options to customize the cluster. The `options` parameter is specified as a JSON object. Some of the values that can be configured:

- `multiPrimary`: Boolean value used to define an InnoDB cluster with multiple writable instances, default to `false`
- `failoverConsistency`: string value indicating the consistency guarantees that the cluster provides, default is `EVENTUAL`
  - Set to `BEFORE_ON_PRIMARY_FAILOVER` to put new queries on hold in the new primary server until after the backlog from the old primary has been applied
- `expelTimeout`: integer value to define the time period in seconds that cluster members should wait for a non-responding member before evicting it from the cluster, default is 0



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

Other options that are available:

- `interactive`
- `disableClone`
- `gtidSetIsComplete`
- `adoptFromGR`
- `memberSslMode`
- `ipWhitelist`
- `groupName`
- `localAddress`
- `groupSeeds`
- `exitStateAction`
- `memberWeight`
- `autoRejoinTries`

The `groupName`, `localAddress`, and `groupSeeds` are advanced options, and their usage is discouraged since incorrect values can lead to Group Replication errors. It is recommended to let AdminAPI to configure them automatically.

A subset of these options can be changed using `cluster.setOption()` function.

## Customizing an Instance

When you add an instance to a cluster using `dba.addInstance(InstanceDef instance, Dictionary options)`, you can provide a list of options to customize the instance. Some of the values that can be configured:

- `recoveryMethod`: the preferred method of state recovery, default is `AUTO`
  - Can be set to `clone` or `incremental`
- `exitStateAction`: state of the instance when it leaves the cluster unexpectedly, default is `READ_ONLY`
  - Can be set to `ABORT_SERVER` (instance shutdown) or `OFFLINE_MODE` (read-only and accepts only connections from clients with administrative privileges)
- `memberWeight`: an integer value with a percentage weight for automatic primary election on failover, default is 50
- `autoRejoinTries`: configures how many times an instance will try to rejoin a cluster after being expelled, default is 0



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

Other options that are available:

- `label`
- `waitRecovery`
- `password`
- `memberSslMode`
- `ipWhitelist`
- `localAddress`
- `groupSeeds`

A subset of these options can be changed using `cluster.setInstanceOption()` function.

## Configuring Secure Connection in a Cluster

- When you set up a cluster using `dba.createCluster()`, the `memberSslMode` option configures SSL connections. The modes are:
  - **AUTO** (the default): SSL encryption is automatically enabled if the server instance supports it or disabled if the server does not support it.
  - **DISABLED**: SSL encryption is disabled for the seed instance.
  - **REQUIRED**: SSL encryption is enabled for the seed instance in the cluster. If it cannot be enabled, an error is raised.
- When you issue the `cluster.addInstance()` and `cluster.rejoinInstance()` commands, the `memberSslMode` option configures SSL connections. The modes are:
  - **AUTO** (the default): SSL encryption is automatically enabled or disabled based on the cluster configuration.
  - **DISABLED**: SSL encryption is disabled for the instance.
  - **REQUIRED**: SSL encryption is enabled for the instance in the cluster.



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

For sandbox instances, `cluster.deploySandboxInstance()` tries to deploy instances with SSL encryption by default where possible.

## Creating a Server Whitelist

- The cluster maintains a list of approved servers that belong to the cluster, known as a whitelist.
  - Only servers in the whitelist can join the cluster.
  - The default whitelist includes the private network addresses that the server has network interfaces on.
- Create your own whitelist with the `ipWhiteList` option to `dba.createCluster()`, `cluster.addInstance()`, or `cluster.rejoinInstance()` to improve security.
  - Pass the IP addresses or subnet CIDR notation as a comma-separated list, surrounded by quotes.
  - Configure the `group_replication_ip_whitelist` system variable on the instance.



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

### Examples of CIDR notations:

- 10.0.0.0/8 includes IP address range 10.0.0.0–10.255.255.255
- 172.16.0.0/16 includes IP address range 172.16.0.0–172.16.255.255
- 192.168.0.0/24 includes IP address range 192.168.0.0–192.168.0.255

## Summary



In this lesson, you should have learned how to:

- Describe MySQL InnoDB cluster and Group Replication
- List typical use cases for MySQL InnoDB cluster
- Contrast the two different modes for deployment
- Configure a MySQL InnoDB cluster
- Administer a MySQL InnoDB cluster



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

## Practices

- 14-1: Creating MySQL InnoDB Cluster
- 14-2: Deploying MySQL Router and Testing the Cluster
- 14-3: Testing High Availability



ORACLE®

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



15

# Conclusion



ORACLE®

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

## Course Goals

In this course, you should have learned how to:

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



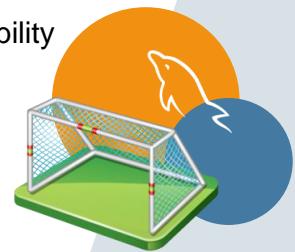
ORACLE®

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

## Course Goals

In this course, you should have learned how to:

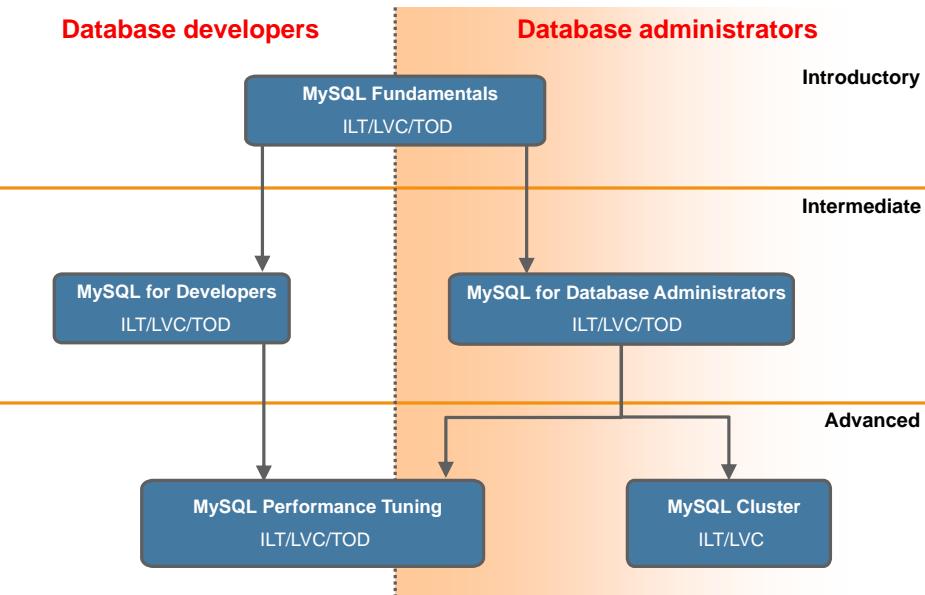
- Maintain a stable system
- Troubleshoot server slowdowns and other common problems
- Identify and optimize poorly performing queries
- Define and implement a backup strategy
- Perform physical and logical backups of your data
- Describe MySQL replication and its role in high availability and scalability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



ORACLE®

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

# Oracle University: MySQL Training



ORACLE®

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

## MySQL Websites

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



ORACLE®

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

## Your Evaluation

- Courses are continually updated, so your feedback is invaluable.
- Thank you for taking the time to give your opinions.



ORACLE®

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

## Thank You

- Congratulations on completing this course!
- Your attendance and participation are appreciated.
- For training and contact information, see the Oracle University website at <http://www.oracle.com/education>.



ORACLE®

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

## Q&A Session

- Questions and answers
- Questions after class
  - Get answers from the online reference manual at <http://dev.mysql.com/doc/mysql/en/faqs.html>.
- Example databases
  - Download the `world`, `employee`, and other sample databases from <http://dev.mysql.com/doc/index-other.html>.



ORACLE®

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