MySQL Performance Tuning

Student Guide

D61820GC20
Edition 2.0
May 2011
D73030

ORACLE

Author

Jeff Gorton

Technical Contributors and Reviewers

Paul Dubois
John Russell
James Day
Kimseong Loh
Mattias Jonsson
Ole Solberg

Editors

Malavika
Richard Wallis

Graphic Designer

Maheshwari Krishnamurthy

Publishers

Jayanthy Keshavamurthy
Veena Narasimhan

Contents

Preface

Profile

Before You Begin This Course

Before you begin this course, you should have the following:

- Experience maintaining a database server (preferably a MySQL server)
- Ability to use MySQL tools to connect to the MySQL Server
- Have knowledge of general SQL statement structure and basic SQL tuning principles
- Experience using the Linux operating systems

Suggested Courses

- D61762GC20: MySQL for Database Administrators

How This Course Is Organized

MySQL Performance Tuning is an instructor-led course featuring lecture, demos, quizzes and hands-on exercises. The practice sessions reinforce the concepts and skills introduced.

Typographic Conventions

Typographic Conventions in Text

The following typographical conventions are used throughout this training guide:

- Computer input and output is printed in this format: Computer input or output. This is also used for the names of executable programs and file locations.
- Keywords from the SQL language appear in this format: `SQL KEYWORD`. SQL keywords are not case sensitive and may be written in any letter case, but the training guide uses uppercase.

When commands are shown that are meant to be executed from within a particular program, the prompt shown preceding the command indicates which command to use. For example, `sys>` indicates a command that you execute from your shell, and `mysql>` indicates a statement that you execute from the `mysql` client program:

```
sys> mysql -u root -h 127.0.0.1
mysql> SELECT * FROM world.City;
```

The "sys" is your command interpreter. On Linux, this is typically a program such as `sh`, `csh`, or `bash`. On Windows, the equivalent program is `command.com` or `cmd.exe`, typically run in a console window. When you enter a command or statement shown in an example, do not type the prompt shown in the example.

Database, table, and column names must often be substituted into statements. To indicate that such substitution is necessary, this manual uses `db_name`, `tbl_name`, and `col_name`. For example, you might see a statement like this:

```
mysql> SELECT col_name FROM db_name.tbl_name;
```

This means that if you were to enter a similar statement, you would supply your own database, table, and column names for the placeholders `db_name`, `tbl_name`, and `col_name`., perhaps like this:

```
mysql> SELECT author_name FROM biblio_db.author_list;
```

In syntax descriptions, square brackets (`[` and `]`) indicate optional words or clauses. For example, in the following statement, `IF EXISTS` is optional:

```
DROP TABLE [IF EXISTS] tbl_name;
```

When a syntax element consists of a number of alternatives, the alternatives are separated by vertical bars (pipe, `|`).

When one member from a set of choices may be chosen, the alternatives are listed within square brackets (`[` and `]`):

```
TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)
```

When one member from a set of choices must be chosen, the alternatives are listed within braces (`{` and `}`):

```
{DESCRIBE | DESC} tbl_name [col_name | wild]
```

Typographic Conventions (continued)

Typographic Conventions in Text (continued)

An ellipsis (. . .) indicates the omission of a section of a statement, typically to provide a shorter version of more complex syntax. For example, `INSERT ... SELECT` is shorthand for the form of `INSERT` statement that is followed by a `SELECT` statement.

An ellipsis can also indicate that the preceding syntax element of a statement may be repeated. In the following example, multiple `reset_option` values may be given, with each of those after the first preceded by commas:

```
RESET reset_option[, reset_option] ...
```

1

Introduction

Course Goals

After completing this course, you should be able to:

- Understand the basics of performance tuning
- Use performance-tuning tools
- Tune the MySQL server instance to improve performance
- Improve the performance of tables based on the storage engine being used
- Implement proper schema design to improve performance
- Improve the performance of MySQL queries
- Describe additional items related to performance tuning

ORACLE

Course Lesson Map

Day 1

1. Introduction to MySQL
2. Performance Tuning Basics
3. Performance Tuning Tools
4. MySQL Server Tuning

Day 2

4. MySQL Server Tuning (continued)
5. MySQL Query Cache

Course Lesson Map

Day 3

6. InnoDB
7. MyISAM
8. Other MySQL Storage Engines and Issues

Day 4

9. Schema Design and Performance
10. MySQL Query Performance
11. Performance Tuning Extras
12. Conclusion

Introductions

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

Classroom Environment

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

MySQL: Overview

- Relational database management system (RDMS) program suite
- World's most popular open source database
  - Fastest growing with over 70,000 downloads per day
- Originally developed by MySQL AB (Sweden)

MySQL is installed on every continent in the world. (Yes, even Antarctica!)

Acquisitions of the MySQL Company

- Sun Microsystems acquired MySQL in 2008.
- Oracle acquired Sun Microsystems in 2010.
- These acquisitions give MySQL:
  - The resources of a major mainstream company
  - Enterprise class support 24×7×365
- All this is combined with strong open source community support.

MySQL Is Powering the World!

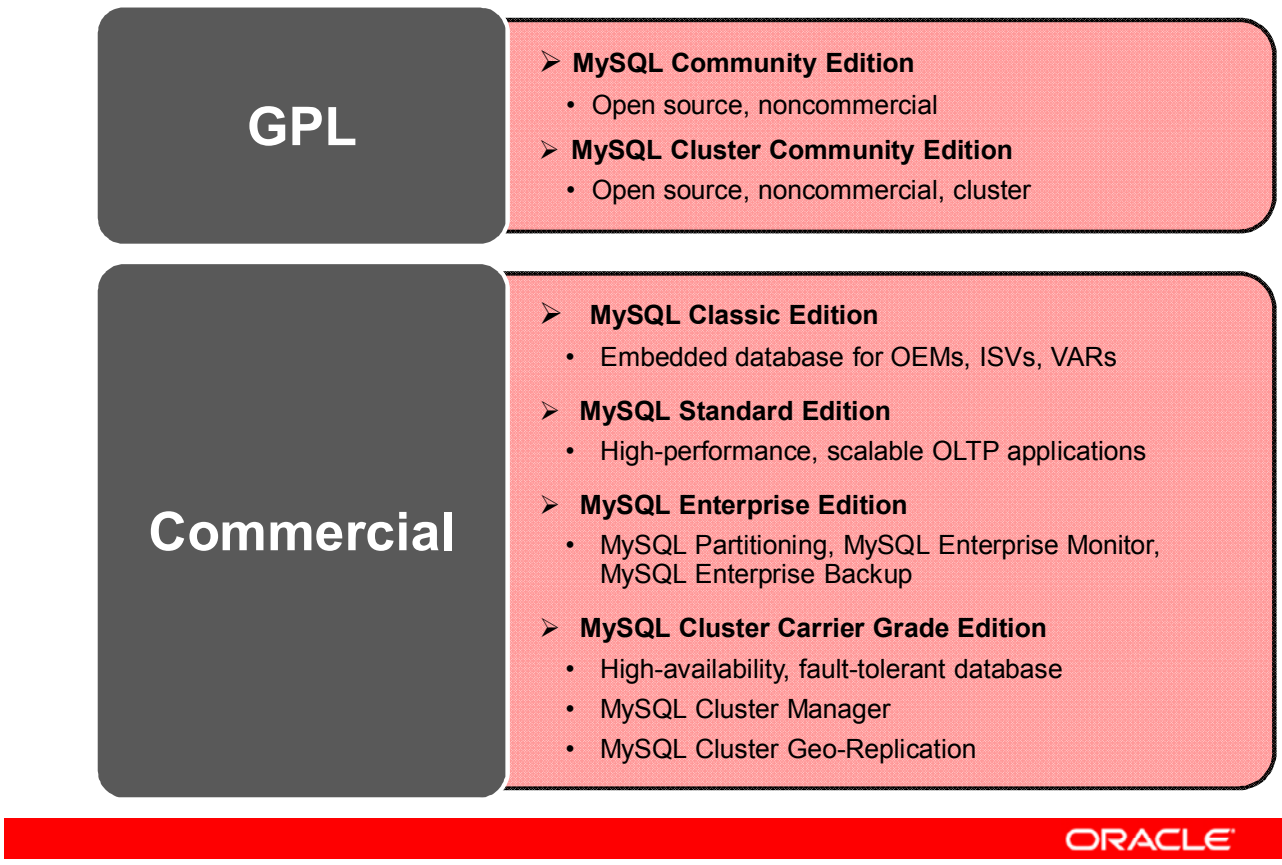
An impressive customer and partner community!

ORACLE

MySQL Is Powering the World!

MySQL's customer and partner base is:

- Centered in five core areas
  - Web 2.0, SaaS, enterprise, telco, and OEM/embedded markets
- Composed of more than 400 blue-chip customers and a wide range of market leaders
- Utilized by virtually every Web 2.0 and e-commerce company in existence
- Strong in the telco market
- Strong in the enterprise market



MySQL Database Server Editions

MySQL Classic Edition is well suited for read-intensive applications.

MySQL Standard Edition can be used for MySQL replication.

In addition to the Community edition, OEMs can license or subscribe to all editions, and end users can subscribe to the Standard, Enterprise, and Cluster Carrier Grade editions.

Find more information about licensing and subscription options, or contact a MySQL sales associate:

- http://mysql.com/products/
- http://www.mysql.com/about/contact/
- https://shop.oracle.com/
  - Select Database from the Product Categories menu, and then select MySQL.

MySQL Tools

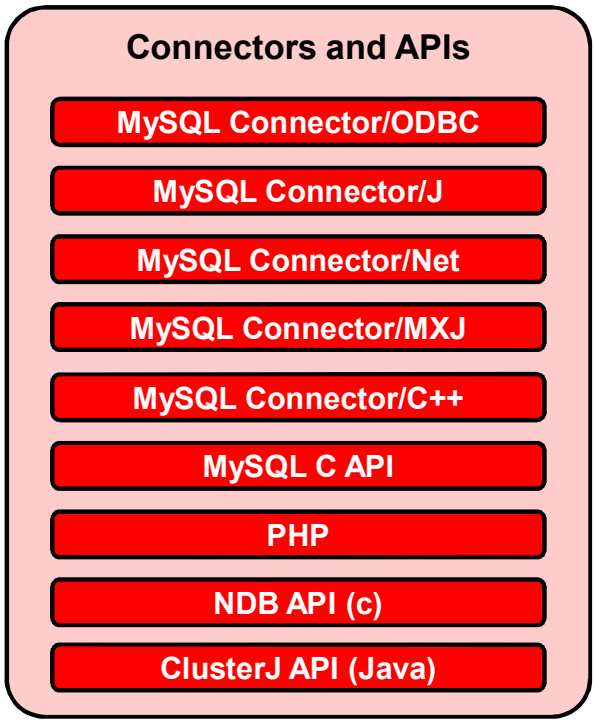
- Graphical user interfaces to your MySQL database
- MySQL Enterprise Monitor
  - Web application that reduces down time, tightens security, and increases throughput
  - Includes MySQL Query Analyzer
- MySQL Enterprise Backup
  - Online hot, nonblocking backups
- MySQL Workbench
  - Visual database design tool that is used to efficiently design, manage, and document databases

ORACLE

MySQL Tools

The MySQL Workbench is available in GPL editions. The other tools are available in commercial editions only.

MySQL Drivers



ORACLE

MySQL Drivers

Third-party connectors that MySQL supports:

- **PHP:** mysqli, ext/mysqli, PDO_MYSQLND, PHP_MYSQLND
  - Provides MySQL connectivity for PHP programs. Currently there are two MySQL-specific PHP extensions available that use libmysql: the mysql and mysqli extensions. There is also MySQL support for the generic PHP Data Objects (PDO) extension. In addition, there is the PHP native driver called mysqlnd, which can replace libmysql in the mysqli extension.
- **Perl:** DBD::mysql
- **Python:** MySQLdb
- **Ruby:** DBD::MySQL, ruby-mysql
- **C++ Wrapper:** For MySQL C API (MySQL++)

You can download connectors and their documentation from the following page:

http://mysql.com/products/connector

MySQL Services

MySQL training

- Comprehensive set of MySQL training courses

MySQL certification

- High-quality certification for MySQL developers and database administrators

MySQL consulting

- Full range of consulting services from startup to optimization

Oracle Premier Lifetime Support

- Offerings to save you time and ensure that you achieve the highest levels of performance, reliability, and uptime

ORACLE

Community Support

- Mailing lists
- Forums
- Community articles
- Planet MySQL blogs
- MySQL Forge
- Twitter

Community Support

- Mailing lists (http://lists.mysql.com/)
- Forums (http://forums.mysql.com)
- Community articles (http://dev.mysql.com/tech-resources/articles)
- Planet MySQL blogs (http://planet.mysql.com/)
- MySQL Forge (http://forge.mysql.com)
- Twitter (http://twitter.com/mysql_community)

Oracle Lifetime Support for MySQL

- 24 × 7 support
- Unlimited support incidents
- Knowledge Base
- Maintenance releases, bug fixes, patches, updates
- MySQL Consultative Support

Oracle Lifetime Support for MySQL

For access to My Oracle Support:

https://support.oracle.com

MySQL: Supported Operating Systems

- Control and flexibility for users
- More than 20 platforms, including:
  - Windows (multiple)
  - Linux (multiple)
  - Solaris
  - Mac OS

MySQL: Supported Operating Systems

For a full list of supported operating systems:

http://www.mysql.com/support/supportedplatforms/database.html

MySQL Websites

- http://dev.mysql.com includes:
  - Developer Zone (Articles, Forums, Forge, Planet MySQL, and more)
  - Downloads (GA and development release)
  - Documentation
- http://www.mysql.com/ includes:
  - Downloads (GA)
  - Services
  - Consulting
  - Resources
  - White papers
  - Webinars

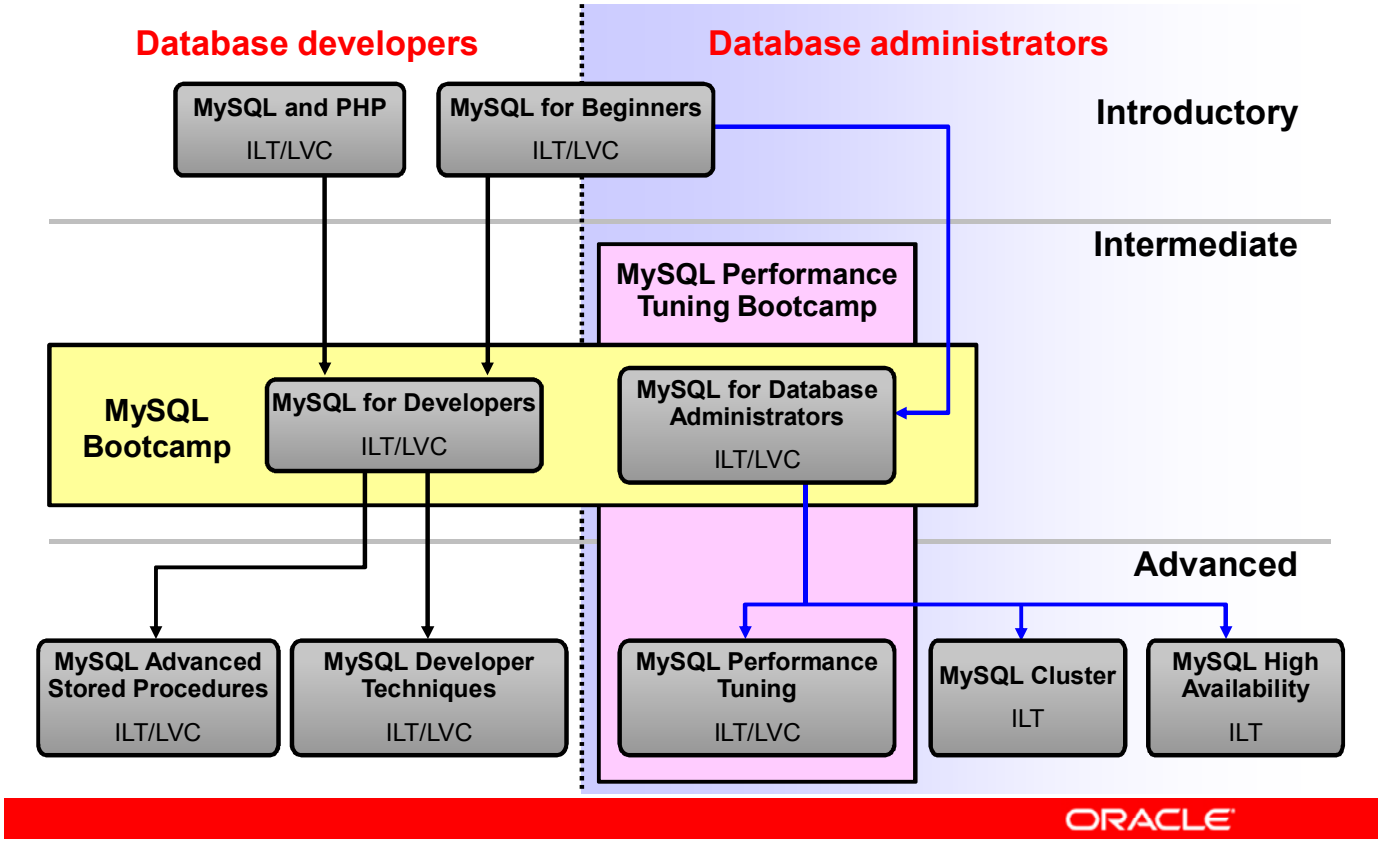
ORACLE

MySQL Websites

For more information:

- http://dev.mysql.com
- http://www.mysql.com/

MySQL Curriculum Footprint



MySQL Curriculum Footprint

Course types:

- **Instructor-Led Training (ILT):** Delivered in a classroom with an instructor and students present at the same location and time.
- **Live Virtual Class (LVC):** Delivered using video and audio through a web-based system (WebEx) in which geographically distributed instructor and students participate, interact, and collaborate in a virtual class environment.

For details about all 18 course products, visit: http://www.oracle.com/education.

MySQL Certifications: Overview

The Oracle Certification Program validates various levels of MySQL expertise:

- **Introductory: Certified Associate**
  - Oracle Certified Associate: MySQL
- **Intermediate: Certified Professional**
  - Oracle Certified Professional: MySQL Database Administrator
  - Oracle Certified Professional: MySQL Developer
- **Advanced: Certified Expert**
  - Oracle Certified Expert: MySQL Cluster Database Administrator

MySQL Certifications: Overview

For complete information about these and other offerings of the Oracle Certification Program:

http://education.oracle.com/certification

MySQL Online Documentation

- *MySQL Reference Manual*
- Topic guides
- Expert guides
- Mysql help tables
- Example databases
- Meta documentation
- Community contributed documentation
- Printed books
- Additional resources

MySQL Online Documentation

http://dev.mysql.com/doc

2

Performance Tuning Basics

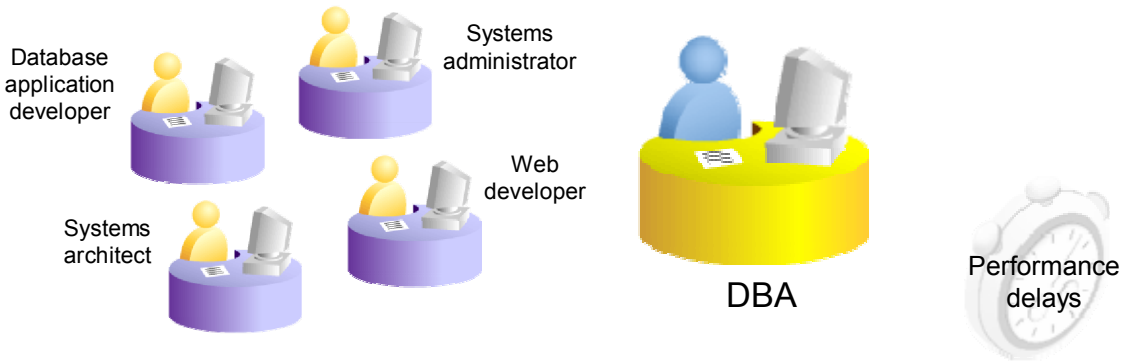
Objectives

After completing this lesson, you should be able to:

- Describe the terms, theories, and strategies associated with performance tuning
- Explain how to implement MySQL deployment guidelines

Thinking About Performance

- Everyone involved with the MySQL database software should think about performance.



- The DBA is the front line for most performance issues.

Thinking About Performance

The database administrator is the starting point for recognizing and correcting the majority of delays in performance. Other system support personnel (database application developers, system administrators, web developers, systems architects, etc.) will be involved in ensuring the best possible performance.

Areas to Tune

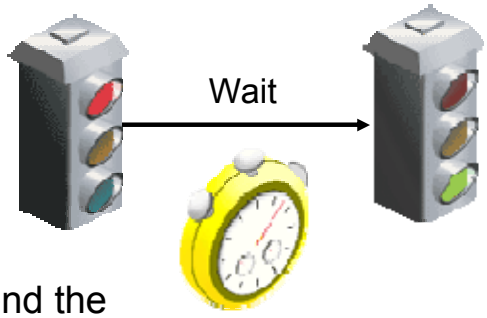
- The MySQL server instance is the primary area that a DBA can tune:
  - Memory
  - Database structure
  - Instance configuration

- Other areas
  - Application tuning
    - SQL statement performance
    - Managing changes and/or increases in the size of the data
  - O/S tuning
    - I/O
    - Swapping
    - Network issues

ORACLE

Performance Tuning Terminology

- Waits
  - The amount of time during which a requested activity is in an inactive state waiting for the request to be processed
- Service times
  - The amount of time needed to send the requested data to the client
- Baseline
  - A basic standard of values that can serve as a comparison for changes on a system



ORACLE

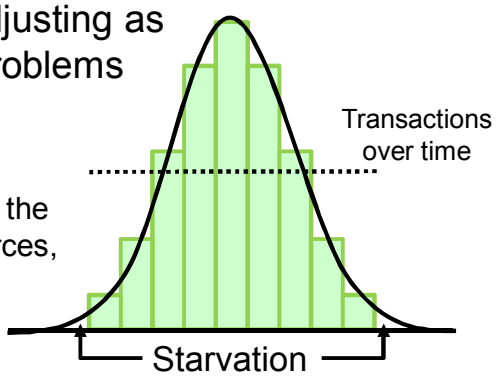
Performance Tuning Terminology

Service times

- Factors affecting the service time include:
  - Network traffic.
  - Processing speed of the CPU.
  - Read/write speed of the disks containing the data.
  - Locks being generated or applied to the data.
- Direct and indirect Measurements:
  - To directly measure the service time, you can sum up all query times for the transactions processed.
  - Considering other, nondirect measures can help you have a more complete picture of the service time:
    - CPU usage.
    - Disk IO latency.
    - Network traffic.
    - Load Average.
    - Number of running queries.

Performance Tuning Terminology

- Throughput
  - The amount of data moved successfully from one place to another in a given time period
  - Monitoring throughput and adjusting as needed can minimize such problems as:
    - Starvation
      - One or more users are using the majority of the system resources, leaving others with limited resources to use.
    - Lack of prioritization
      - One or more users may have business requirements that make their requests higher priority than others.



Performance Tuning Terminology (continued)

Throughput

- The metric for throughput is the number of transactions per a given time (second/min/hour).
  - For all the transactions that are processed, only some transactions from the mix can be counted in the metric results.
  - If you are administering MySQL in interactive multiuser applications, it is necessary to monitor throughput metrics and adjust your system based on the results.
    - One example of a multiuser application is an online transaction processing (OLTP) system.

Performance Tuning Terminology

- Response time and latency
  - Response time: Total time it takes from when a user makes a request until a response is received
  - Latency: Delay incurred in communicating a message
  - Two ways to develop metrics for response time and latency:
    - Average/minimum/maximum
    - 90th percentile
- Scalability
  - Ability of a system to adapt to increased demands

ORACLE

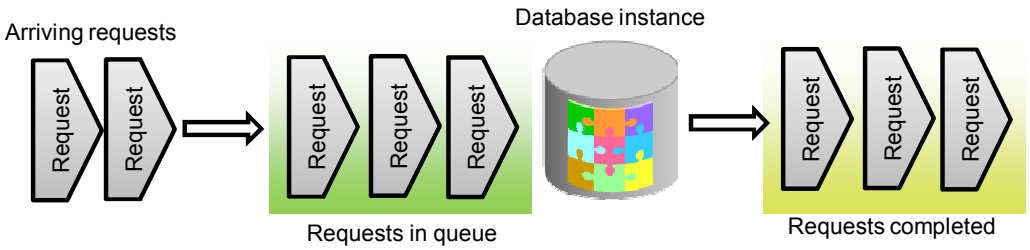
Performance Tuning Terminology (continued)

Response Time / Latency

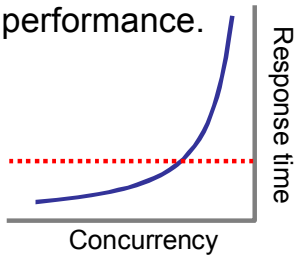
- Both response time and latency are usually discussed together due to the nature of their interaction.
- Metrics:
  - **Average/Minimum/Maximum:** The metric for response time and latency can be based on the average, minimum and maximum response times (millisecond/second/min) for a transaction to process.
  - **90th Percentile:** The metric for response time and latency can be based on 90th Percentile response time (millisecond/second/min) for each group of transactions.
    - In a set of 100 values that are sorted from best to worst, the 90th percentile simply means the 90th value in the list.
- You must consider the following issues when using response time and latency metrics:
  - **Count wall clock time:** These metrics are dependent upon a number of clock ticks instead of an arbitrary time, the actual time for a transaction response time might vary between the same transactions are executed if the clock rate differs.
  - **Who cares what else is happening:** These metrics do not take into account what else is happening on the system when the transactions are executed.

Performance Tuning Terminology

- Queuing theory
  - A model for evaluating the efficiency of a system that consumes multiple resources



  - Applying this model to databases: Use response times and throughput metrics to evaluate and improve performance.
  - "Hockey stick" phenomenon
    - In systems where the queue is growing faster than the system is able to handle the transactions (saturation), the end user experiences an exponential delay.



ORACLE

Performance Tuning Terminology (continued)

"Hockey Stick" Solutions

- When you are attempting to find a solution for the "Hockey Stick" phenomenon, it is important for you to look at both the delay directly related to the queue and the service time associated with the processing of the transactions.
  - If you improve one, you reduce the user response time.
  - Improving both is the optimal solution.

Performance Tuning Terminology

- Benchmarking
  - Testing designed to compare the qualities or performance of different systems, hardware, or processes
  - Benchmarks are a great tool to:
    - Quantify application performance: Use benchmarks to develop baselines for performance.
    - Measure performance effect of the changes: To determine if a change has a positive effect on performance, perform a benchmark before and after the change.
    - Validate scalability: Determine the effect on performance of adding systems to an existing architecture.
    - Plan deployment: Determine the performance of different deployment options.

ORACLE

Performance Tuning Terminology (continued)

Benchmarking

- Database benchmarks are designed to assess the relative performance of your applications ability to perform transactions in a timely and efficient manner.

Quiz

What are the ways to develop metrics for response time and latency? (Select two of the following.)

a. Average/minimum/maximum
b. Hockey stick
c. Queuing theory
d. 90th percentile

Answer: a, d

Benchmark Planning

- Replicate the real-world application as closely as possible.
  - A benchmark is a measure of a snapshot of your total system.
  - Consider using a log of transactions recorded from your live system in the simulated environment.
- Use a representative data population.
  - Ensure that the data population you study is representative of the total data population of your system.
- Use real database size.
  - Use a database size that is equivalent to the data in your system.
- Use real input values.
  - Use data that is equivalent to the real data in your system.

ORACLE

Benchmark Planning

- Use a similar number of connections.
  - Execute benchmarks with the same number of concurrent connections that your real system must handle.
- Use a similar “think time.”
  - Think time: A average time a user spends looking at a webpage before determining the next action
  - The time required to load the page is not considered in this step.
- Understand the effects of caching.
  - Ensure that caching is cleared to prevent repetitive data from producing inaccurate results.
- Use similar server settings, hardware, O/S, network, and so on.

ORACLE

Benchmark Errors

- Comparatively different database sizes
  - Do not test with 1 GB of data if the production server holds 100 GB of data.
- Using a uniform distribution that is not natural to the real world
  - Use data that contains a similar representation of the transactions that your production server faces.
- Testing in a single-user scenario
  - If your production server has multiple connections made to it, a benchmark is not accurate if it is set up as a single-user scenario.
- Minimizing (or not using) think times

ORACLE

Benchmark Errors

Uniform distribution example

- If you maintained a bookseller site, the "Harry Potter" book series would be ordered more frequently than the "Zulu Dictionary".

Benchmark Errors

- Benchmarking on a single host
  - Connect to the MySQL server from numerous hosts.
- Executing the same queries in a loop (query cache issues)
  - Simulate queries that are unique to avoid the query cache affecting results.
- Ignoring server warm-up
  - Let your queries run on the benchmark server for a few hours before you collect results.
- Using default MySQL server settings
  - Make adjustments to the MySQL default server settings so that the settings are more appropriate for your system.

Tuning Steps

- Check the systems:
  - O/S and general machine health should be the first areas to check.
- Tune first the areas that have the most potential for gain:
  - Longest waits
  - Longest service times
- Tune to a goal:
  - Stop tuning when the goal is met.
  - Tuning goals should be specific, measurable, and achievable.

General Tuning Session

1. Define the problem and state the goal.
2. Collect current performance statistics.
3. Consider some common performance errors.
4. Build a trial solution.
5. Implement and measure the change.
6. Evaluate the solution.
   - If the solution meets the goal, define the new baseline.
   - If the solution does not meet goal, consider other common performance errors and then retune.

ORACLE

Deploying MySQL

- Use automation whenever possible.
- Have different servers for different systems.
  - Database and web servers have different configuration, quality, and scaling requirements.
- Have MySQL on an internal network.
  - A MySQL server should never be directly accessible to the Internet.
  - MySQL servers that are accessible via a LAN are acceptable if the LAN is separated from the Internet by a firewall (preferably a hardware firewall).
  - SSH tunneling should be used to give access to a MySQL server remotely.

ORACLE

Deploying MySQL

- Design and implement a regular backup schedule.
- Use binary logging to enable point-in-time recovery.
- Enable the slow log to catch slow queries.
- Monitor the entire infrastructure (systems and network) to have a holistic and historical view of it.
  - Performance problems are not always related to the MySQL server.

Quiz

Which of the following is not a typical benchmark error?

a. Connecting to the MySQL server from multiple hosts
b. Minimizing (or not using) think times
c. Ignoring server warm-up
d. Using default MySQL server settings

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe the terms, theories, and strategies associated with performance tuning
- Explain how to implement MySQL deployment guidelines

3

Performance Tuning Tools

Objectives

After completing this lesson, you should be able to:

- Use MySQL tools to monitor the MySQL server
- Use tools to create benchmarks in the MySQL server environment
- Use tools to create a workload that stresses the MySQL server

MySQL Monitoring Tools

- SHOW STATUS
- information_schema
  - The information database
- performance_schema
  - The performance-monitoring database
- SHOW OPEN TABLES

MySQL Monitoring Tools

SHOW STATUS command

- Displays server status information for your MySQL server
  - WHERE and LIKE clauses can be used with this tool.
  - Status values for all connection to MySQL can be obtained with the GLOBAL modifier.
- http://dev.mysql.com/doc/refman/5.5/en/show-status.html

information_schema database

- Stores information about all the other databases that your MySQL server maintains
- http://dev.mysql.com/doc/refman/5.5/en/information-schema.html

performance_schema database

- Stores low-level performance data related to the execution of your MySQL server
- http://dev.mysql.com/doc/refman/5.5/en/performance-schema.html

SHOW OPEN TABLES command

- Displays the the non-TEMPORARY tables that are currently open in the table cache
  - FROM, WHERE, and LIKE clauses can be used with this tool.
- http://dev.mysql.com/doc/refman/5.5/en/show-open-tables.html

MySQL Monitoring Tools

- `SHOW ENGINE INNODB STATUS`
- `SHOW PROCESSLIST`
- `mysqladmin`
  - Client for performing administrative operations
  - `debug` command: Tells the server to write debug information to the error log
- MySQL Enterprise Monitor
  - Virtual MySQL DBA assistant

MySQL Monitoring Tools (continued)

`SHOW ENGINE INNODB STATUS` command

- Displays extensive information from the standard InnoDB Monitor about the state of the InnoDB storage engine
- http://dev.mysql.com/doc/refman/5.5/en/innodb-monitors.html

`SHOW PROCESSLIST` command

- Displays the MySQL threads that are active on your server
- http://dev.mysql.com/doc/refman/5.5/en/show-processlist.html

`mysqladmin` client

- http://dev.mysql.com/doc/refman/5.5/en/mysqladmin.html

MySQL Enterprise Monitor

- A commercial product that continuously monitors your MySQL servers and alerts you to potential problems before they impact your system
- http://www.mysql.com/products/enterprise/monitor.html

Practice 3-1 Overview: MySQL Monitoring Tools

In this practice, you compare the available MySQL monitoring tools by:

- Using performance monitoring commands that are available within the `mysql` client
  - This includes using various `SHOW` statements along with built-in information and performance databases.
- Using the `mysqladmin` client
- Setting up and reviewing the information that can be obtained from the MySQL Enterprise Monitor

ORACLE

Quiz

What is the Oracle commercial product that continuously monitors your MySQL servers and alerts you to potential problems before they affect your system?

a. `mysqladmin`
b. Oracle Beehive
c. MySQL Enterprise Monitor
d. MySQL Cluster

Answer: c

Open Source Community Monitoring Tools

- dim_STAT
  - Created by Dimitri Kravtchuk
- mysqlresources
  - Perl script created by Giuseppe Maxia
- Maatkit
  - Created by Daniel Nichter and Baron Schwartz

Open Source Community Monitoring Tools

dim_STAT application

- A high-level and detailed monitoring and performance analysis tool for Solaris and Linux systems
- http://dimitrik.free.fr

mysqlresources application

- A Perl script that finds the resources (CPU, memory, and file descriptors) used by your MySQL process
- http://www.oreillynet.com/databases/blog/2006/07/measuring_resources_for_a_mysq_1.html

Maatkit toolkit

- A toolkit you can use to prove that replication is working correctly, fix corrupted data, automate repetitive tasks, speed up your servers, and much more
- http://www.maatkit.org

Open Source Community Monitoring Tools

- mysqlreport
  - Created by Daniel Nichter
- mytop
  - Created by Jeremy Zawodny
- Preinstalled Linux tools
  - `iostat`
  - `vmstat`
  - `sar`
  - `top`

ORACLE

Open Source Community Monitoring Tools (continued)

mysqlreport application

- A tool that provides you with an easy-to-read report of important MySQL status variables transformed from the `SHOW STATUS` values
- http://hackmysql.com/mysqlreport

mytop application

- A console-based (non-GUI) tool for monitoring the threads and overall performance of your MySQL server
- http://jeremy.zawodny.com/mysql/mytop

Linux tools

- **`iostat`:** Provides valuable I/O information about your server, such as the amount of I/O being processed per device, the type of process, the request size of the process, and the queue size and response time for the process
- **`vmstat`:** Provides a high-level view of your system load

- **`sar`:** Collects, reports, and saves your system activity information for historical comparison
- **`top`:** Provides a dynamic real-time view of your running system to include system summary information as well as a list of tasks currently being managed by the Linux kernel

Practice 3-2 Overview: Open Source Community Monitoring Tools

In this practice, you compare the available open source community monitoring tools by:

- Starting and using the dim_STAT server and service
- Reviewing the mysqlresources application output
- Reviewing the multiple tools available in the Maatkit toolkit and using a select number of these tools
- Executing the mysqlreport application, performing multiple queries against the MySQL server, and executing the mysqlreport application again
- Starting the mytop application and performing a select number of commands in the application

ORACLE

Quiz

Which of the following open source community monitoring tools contain applications that you can use to prove replication is working correctly, fix corrupted data, automate repetitive tasks, and speed up your servers?

a. dim_STAT
b. Maatkit
c. mysqlresources
d. mytop

Answer: b

Benchmark Tools

- MySQL benchmark tools
  - sql-bench
  - mysqlslap
- Open source community benchmark tools
  - DBT-2
    - Created by Craig Thomas and Mark Wong
  - SysBench
    - Maintained by Alexey Kopytov

Benchmark Tools

sql-bench application

- A set of Perl applications that you can use to measure the overall performance of the server. You can use the results as a basis for comparison.
- http://dev.mysql.com/doc/refman/5.5/en/mysql-benchmarks.html

mysqlslap application

- A diagnostic program designed to emulate client load for a MySQL server and to report the timing of each stage
- http://dev.mysql.com/doc/refman/5.5/en/mysqlslap.html

DBT-2 application

- An online transaction processing (OLTP) transactional performance test that enables you to simulate a wholesale parts supplier database.
  - The simulation includes several workers accessing the database, updating customer information, and checking parts inventories.
  - The results of a test run include transactions per second, CPU utilization, I/O activity, and memory utilization.
- http://osdldbt.sourceforge.net

SysBench application

- A modular, cross-platform, and multithreaded benchmark tool you can use to evaluate the O/S parameters that are important for a system running a database under intensive load
- http://sysbench.sourceforge.net

Stress Tools

- db_STRESS
  - Created by Dimitri Kravtchuk
- Apache JMeter
- Benerator
  - Created by Databene
- stress_driver
  - Maintained by Danny Faught

Stress Tools

db_STRESS application

- A tool that enables you to put a load on your database system that produces a high-level metric (TPS: transactions per second)
- http://dimitrik.free.fr/db_STRESS.html

Apache JMeter application

- A tool that you can use to simulate a heavy load on your server, network, or object to test its strength or to analyze overall performance under different load types
- http://jakarta.apache.org/jmeter/

Benerator application

- A performance test data–generation tool that you can use to completely synthesize test data or import and anonymize (remove personal data) your own production data
- http://databene.org/databene-benerator

stress_driver application

- A general-purpose stress-test tool using the Perl scripting language
- http://stress-driver.sourceforge.net

Practice 3-3 Overview: Benchmark Tools

In this practice, you compare the available benchmarking tools by:

- Running all the tests of the sql-bench benchmark suite against the MySQL server
- Preparing and running the mysqlslap benchmark tool using a predefined script
- Preparing and running the sysbench benchmark tool using built-in tests against the MySQL server

ORACLE

Quiz

Dimitri Kravtchuk is an open source developer who has created multiple tools to assist MySQL database administrators in their everyday tasks. At which of the following website locations is his main page?

a. http://www.dimitrik.com
b. http://hackmysql.com
c. http://www.maatkit.org
d. http://dimitrik.free.fr

Answer: d

Summary

In this lesson, you should have learned how to:

- Use MySQL tools to monitor the MySQL server
- Use tools to create benchmarks in the MySQL server environment
- Use tools to create a workload that puts stress on the MySQL server

4

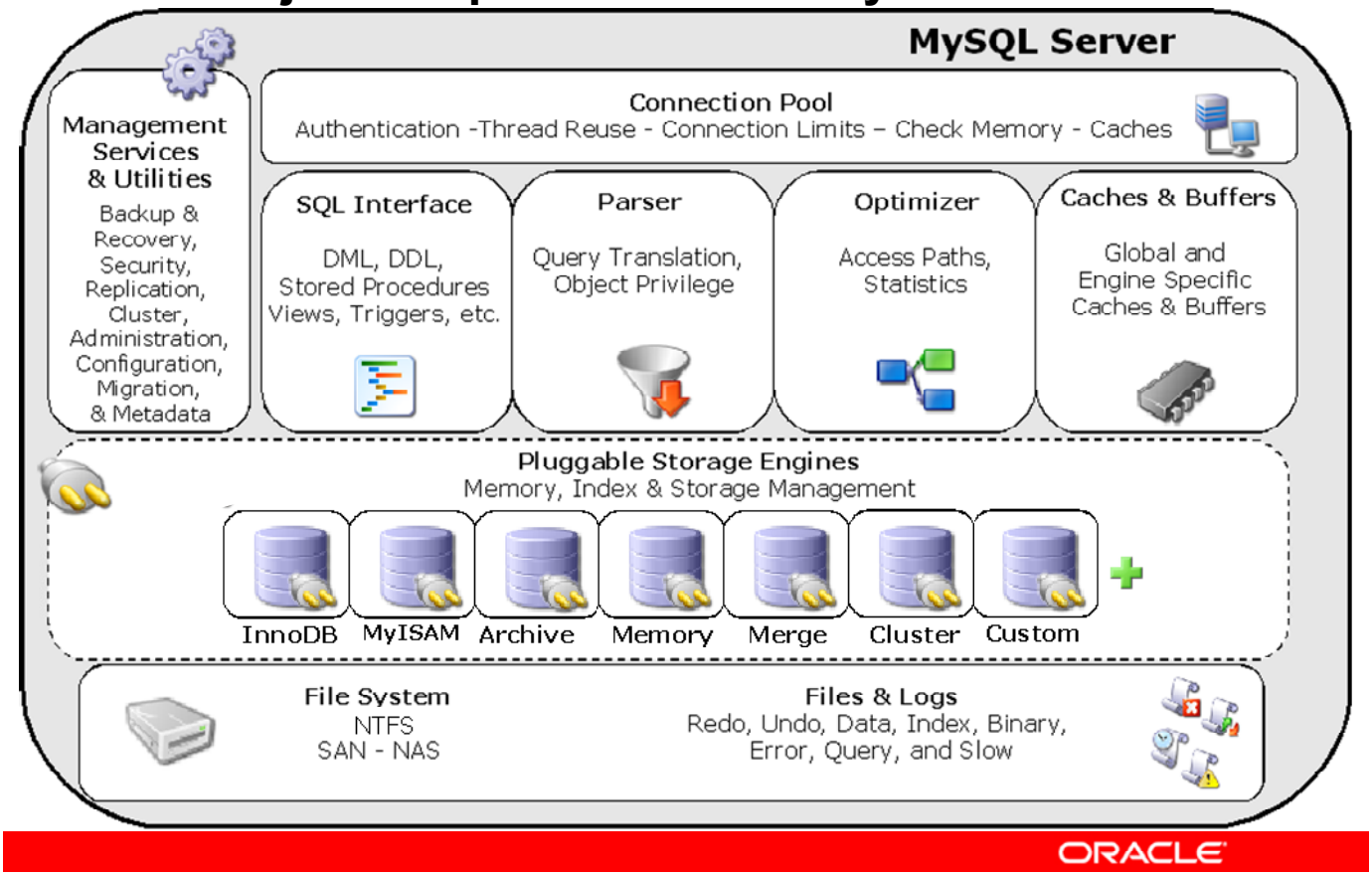
MySQL Server Tuning

Objectives

After completing this lesson, you should be able to:

- Describe the advantages and limitations of the MySQL server architecture that directly affect performance
- Use server settings and status variables to evaluate and alter the performance of the MySQL server

Major Components of the MySQL Server



Major Components of the MySQL Server

The slide shows the following MySQL server elements:

- Connection pool
- SQL interface
- Parser
- Optimizer
- Caches and buffers
- Pluggable storage engines
- File system
- Files and logs
- Management services and utilities

Quiz

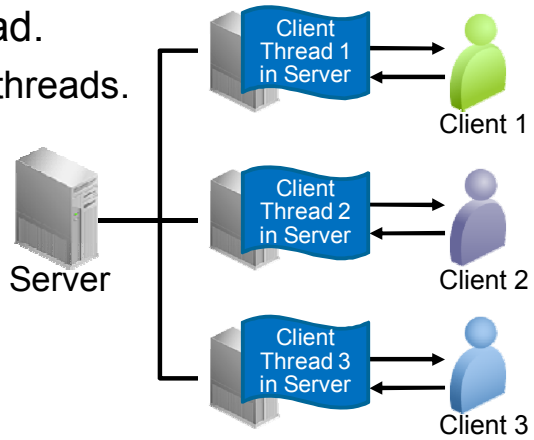
Which of the following is *not* a function of the MySQL server connection pool?

a. Authentication
b. Connection limits
c. Query translations
d. Thread reuse

Answer: c

MySQL Thread Handling

- MySQL is a multithreaded server.
- Single process; multiple threads
- Each connection has its own thread.
  - 1,000 connections require 1,000 threads.
- Helper threads may be used.
  - Connection manager threads
  - Signal threads (alarms)
  - InnoDB read and write threads
  - Event scheduler
- Thread cache capability
  - Thread caching allows unused threads to remain in memory for reuse.



MySQL Thread Handling

Single process, multiple threads

- 32-bit system limits the usable memory; 64-bit allows a much higher limit.
- Good thread support needs to be considered when you choose an O/S.

Connection manager threads

- **TCP/IP connection request**: On Unix, this manager thread also handles Unix socket file connection requests.
- **Windows only:**
  - Shared-memory connections
  - Named-pipe connections

    - MySQL creates a thread for named pipes and shared memory only if the server supports them.

Thread caching can make a big difference.

- Threads that are no longer needed are released from memory.
- Thread caching allows unused threads to remain in memory to be used by a new connection.

Quiz

Thread caching allows unused threads to remain in memory for reuse.

a. True
b. False

Answer: a

MySQL Memory Usage

- MySQL server code uses a minimal amount of memory.
- You should review and adjust (as needed) the following global buffers system variables to improve memory usage:
  - `key_buffer_size`, `query_cache_size`, `innodb_buffer_pool_size`, `table_cache`
- Threads are per connection and should have a reasonable size thread stack.
- Kernel objects
  - Sockets, kernel stacks, file descriptor table, and file system cache

MySQL Memory Usage

- The following system variables should be reviewed and adjusted as needed to improve memory usage per thread: `sort_buffer_size`, `tmp_table_size`, and `read_buffer_size`
- Refer to the MySQL 5.5 documentation to obtain a better understanding of how MySQL uses threads for client connections: http://dev.mysql.com/doc/refman/5.5/en/connection-threads.html

Quiz

Which of the following is *not* a kernel object?

a. File descriptor table
b. File system cache
c. Sockets
d. Storage engine

Answer: d

Simultaneous Connections in MySQL

- Determined by the `max_connections` system variable
- Things to consider when setting this value:
  - Quality of the thread library
  - RAM availability
  - Amount of RAM used for each connection
  - Workload required from each connection
  - Desired response time
- O/S limitations:
  - Linux/Solaris: 500 to 1,000 connections is average
  - Windows: 16,384 by default.
- `Max_used_connections` status variable
  - Can assist in determining optimal setting

Simultaneous Connections in MySQL

`max_connections (151)`

- The server adds one additional connection to any value for use by accounts that have the `SUPER` privilege.
  - Can be used by administrators to connect and diagnose problems even if the maximum number of connections is reached

O/S limitations

- Linux and Solaris:
  - Up to 10,000 connections are possible if there is a large amount of RAM (multiple gigabytes), the workload from each connection is relatively small, and the desired response time is reasonable.
- Windows:
  - The default maximum number of open files that MySQL on Windows can support is 16,384 (can be increased by setting `--open-files-limit` at server startup).
  - The number of simultaneous connections to set in Windows can be calculated by multiplying the number of open connections by 2, and then adding to this product the number of open connections:
    (2 × number of open tables) + number of open connections

`Max_used_connections`

- This status variable indicates the maximum number of connections that MySQL has had open at the same time since the server was last restarted.
- This value provides a benchmark to help you decide the maximum number of connections that your server should support.
- It can also help in traffic analysis.

Quiz

Which status variable can assist in determining the optimal `max_connections` server setting for your system?

a. `key_buffer_size`
b. `Max_used_connections`
c. `query_cache_size`
d. `table_cache`

Answer: b

Reusing Threads

- `thread_cache_size` determines the number of threads that the server can store for reuse.
- Things to consider when setting this value:
  - Review `Connections` and `Threads_created`.
  - Calculate thread pool effectiveness.

```
100 - ((Threads_created/Connections) * 100)
```

- You should evaluate these settings against the time that the server has been up and running, or since the status variables have been flushed.
  - `Connections/Uptime`
  - `Threads_created/Uptime`
  - `FLUSH STATUS` resets most counters to zero.

ORACLE

Reusing Threads

`thread_cache_size` system variable

- Set this system variable to cover fast variations.
  - For servers that vary from 50 to 150 active connections, set this system variable to `100`.
  - Setting this system variable to accommodate rare bursts of connections at one time (such as `500` connections for a system that typically sees 50 to 150 active connections) is a waste of memory.
  - Increase the `back_log` system variable from the default (`50`) to `200` in cases where connections are being requested faster than the O/S and MySQL can process them.
- By examining the difference between the `Connections` and `Threads_created` status variables, you can assess the efficiency of the thread cache performance.
  - The `Connections` status variable lists the number of connection attempts (successful or not) to the MySQL server.
  - The `Threads_created` status variable lists the number of threads created to handle connections.

`Uptime` status variable

- The number of seconds the server has been up and running

`Uptime_since_flush_status` status variable

- The number of seconds the server has been running since the last `FLUSH STATUS` statement was issued

Thread Cache: Scenario

Given the following values, what is the thread cache effectiveness?

```
thread_cache_size = 30    Threads_created = 29
                          Connections = 29000
                          Uptime = 1000
                          Uptime_since_flush_status = 345678
```

Is the `thread_cache_size` system variable set appropriately for this server? Why?

Thread Cache: Scenario

- Using the calculation 100 – [(`Threads_created` / `Connections`) * 100] can tell you the thread cache hit rate for your server. For this particular scenario, the thread cache hit rate is 99.9%.
- Using the calculation `Connections/Uptime` can tell you the number of connections made per second. For this particular scenario, an average of 29 connections are made per second.
- Using the calculation `Threads_created/Uptime` can tell you the number of threads that needed to be created per second. For this particular scenario, an average of two threads are created per second.
- **Conclusion:** The number of threads being created per second is acceptable. You can increase or decrease the `thread_cache_size` system variable a small amount and check whether there are any peaks of high thread creation rates. If the values remain relatively similar, you can set the `thread_cache_size` system variable to 0 and use the memory savings for another setting.

Practice 4-1 Overview: Effects of Thread Caching

In this practice, you execute a number of random queries against the MySQL server by using multiple connections to determine the best setting for the `thread_cache_size` system variable. The practice covers the following topics:

- Simulating an environment where 128 connections result in approximately 900 transactions running against the MySQL server every second
- Modifying the `thread_cache_size` system variable, simulating the environment, and then calculating the thread cache hit rate

ORACLE

Reusing Tables

The `table_open_cache` system variable determines the number of cached open tables that the server can allow.

- With the MySQL server being multithreaded, multiple connections may be running against a table at one time, and each of these threads opens a table.
- Setting this value too low:
  - The MySQL server has to constantly send disk I/O requests to open the required tables.
- Setting this value too high:
  - Your server can max out its RAM, which can lead to connection refusals, failure of queries to execute, or general server problems.

ORACLE

Reusing Tables

- The `table_open_cache` system variable sets the number of open tables for all threads.
- This system variable is temporarily extended when the cache is full and a new table needs to be opened.
  - Once the table that caused the extended state is no longer being used, the table is closed and released from the cache.

Setting `table_open_cache`

- Determine the proper `table_open_cache` setting:
  - `max_connections` system variable
    - Configures the number of simultaneous client connections that the MySQL server allows
  - `Open_tables` status variable
    - Displays the number of tables currently open
  - `Opened_tables` status variable
    - Displays the number of tables that have been opened
  - Calculate the effectiveness of the `table_open_cache` server setting:

    (`Open_tables` / `Opened_tables`) * 100
- Other factors:
  - `FLUSH TABLES`: Closes all open tables

ORACLE

Setting `table_open_cache`: Scenario 1

Given the values in the following example, what can you conclude about the `table_open_cache` setting?

```
table_open_cache = 600    Open_tables = 112
max_connections = 151     Opened_tables = 1712
                          Uptime = 14325067
                          Uptime_since_flush_status = 5045067
```

- How long has the server been up and running?
- How long has the server been up and running since the status variables were reset?
- How often did the server have to open a table?
- What is significant about the number of current open tables in comparison to the table cache?
- Is the `table_open_cache` setting too high, too low, or just right?

ORACLE

Setting `table_open_cache`: Scenario 1

- The `Uptime_since_flush_status` status variable tells you that the server has been running for approximately 59 days since the `FLUSH STATUS` statement was executed. The number of `Opened_tables` tells you that MySQL had to open a table (not in the cache) only about once every hour. The number of current open tables is approximately 1/5th of the table cache setting. We can conclude that the `table_open_cache` setting is probably too high.
- Why is the `Opened_tables` status variable larger than the `table_open_cache` setting?
  - The `Opened_tables` status variable tells you the number of times the MySQL server has opened tables to handle query executions. The `table_open_cache` setting tells the MySQL server how many open tables can be managed simultaneously for all open threads at one time. Since the `Opened_tables` status variable is simply a counter to the number of times the server had to open a table to handle a query execution, this number can be higher than the `table_open_cache` setting.
  - To learn more about how MySQL opens and closes tables, refer to http://dev.mysql.com/doc/refman/5.5/en/table-cache.html.

Setting `table_open_cache`: Scenario 2

Given the values in the following example, what can you conclude about the `table_open_cache` setting?

```
table_open_cache = 128    Open_tables = 128
max_connections = 151     Opened_tables = 678
                          Uptime = 2459870
                          Uptime_since_flush_status = 1349870
```

- How long has the server been up and running?
- How long has the server been up and running since the status variables were reset?
- How often did the server have to open a table?
- What is significant about the number of current open tables in comparison to the table cache?
- Is the `table_open_cache` setting too high, too low, or just right?

Setting `table_open_cache`: Scenario 2

- The `Uptime_since_flush_status` status variable tells you that the server has been running for approximately 16 days since the `FLUSH STATUS` statement was executed. The number of `Opened_tables` tells you that MySQL had to open only two tables (not in the cache) approximately once every hour. The number of current open tables is maxed out. We can conclude that the setting for the `table_open_cache` system variable is set correctly for this server.

Setting `table_open_cache`: Scenario 3

Given the values in the following example, what can you conclude about the `table_open_cache` setting?

```
table_open_cache = 64      Open_tables = 130
max_connections = 151      Opened_tables = 1560160
                           Uptime = 250576
                           Uptime_since_flush_status = 44576
```

- How long has the server been up and running?
- How long has the server been up and running since the status variables were reset?
- How often did the server have to open a table?
- What is significant about the number of current open tables in comparison to the table cache?
- Is the `table_open_cache` setting too high, too low, or just right?

ORACLE

Setting `table_open_cache`: Scenario 3

- The `Uptime_since_flush_status` tells you that the server has been running for approximately half a day since the `FLUSH STATUS` statement was executed. The value of the `Opened_tables` status variable tells you that MySQL had to open up approximately 35 tables per second and the number of current open tables is greater than the `table_open_cache` system variable.
- Increasing `table_open_cache` on this system will reduce the performance bottleneck that this server is facing.
- Another factor to consider is how fast the value of the `Opened_tables` status variable is increasing. If the value of `Opened_tables` is increasing rapidly, the `table_open_cache` setting is too low; increasing it can improve performance of this server by reducing the rate that tables are open. If the value of `Opened_tables` is not increasing rapidly, the `table_open_cache` setting should remain where it is and evaluated at a later time.

Practice 4-2 Overview: Table Caching

In this practice, you execute a sampling of queries from your logs against the MySQL server to determine the best setting for the `table_open_cache` system variable. The practice covers the following topics:

- Executing the `mysqlslap` application to simulate 20 separate connections that execute a sampling of 200 queries against the `employees` database
- Modifying the `table_open_cache` system variable, simulating the environment, and then calculating the effectiveness of the `table_open_cache` setting

ORACLE

Quiz

The `table_open_cache` system variable can be temporarily extended when the cache is full and a new table needs to be opened.

a. True
b. False

Answer: a

Managing Files

- The `open_files_limit` system variable determines the maximum number of file descriptors the server can use.
  - When the MySQL server requires more than the allotted file descriptors allowed by this setting, the server displays an error stating that there are too many open files.
- `Open_files`
  - This status variable displays the number of files that the MySQL server has currently open.
- `Opened_files`
  - This status variable displays the number of files that have been opened.

ORACLE

Managing Files

`open_files_limit`

- When the MySQL server requests more file descriptors than the O/S can accommodate, the server writes a warning to the error log.

`Open_files`

- Files currently open by the storage engines themselves (or other nonserver processes such as sockets or pipes) are not counted against this status variable.

`Opened_files`

- The count is for files that were opened when the server issued a `my_open()` command. Not all server functions that open files use this command.

Setting File Handles

Set the Linux O/S number of file handles:

- Verify that the O/S maximum number of file handles is set properly.
- A reasonable number is 256 for each 4 MB of RAM.
- Updating the Linux O/S.
  - View the current `file-max` limits.
  - Update the `file-max` limits.
  - Reload the changes.

Setting File Handles

- Before you set the number of file descriptors that MySQL can use, it is important for you to make sure that the O/S maximum number of file handles is set properly.
- If you have a 2 GB system, you need to set the number of file descriptors to 128,000 (2 GB/4 MB * 256).
- Updating the Oracle Enterprise Linux O/S:
  1. View current `file-max` limits.
     ```
     sysctl fs.file-max
     ```
  2. Update `file-max` limits.
     ```
     sysctl -w file-max=128000
     ```
  3. Reload `/etc/sysctl.conf` without having to restart the server.
     ```
     sysctl -p
     ```

Practice 4-3 Overview: Setting `open_files_limit`

In this practice, you set the `open_files_limit` system variable based on the operating system settings. The practice covers the following topics:

- Reviewing the memory details for the hardware being used
- Reviewing the number of file descriptors that the O/S allows to be open concurrently
- Updating the `open_files_limit` system variable equal to the `fs.file-max` setting for the server on which MySQL is running

ORACLE

Quiz

Which system variable determines the maximum number of file descriptors that the server can use?

a. `open_files_limit`
b. `Open_files`
c. `Opened_files`
d. `file-max`

Answer: a

Practice 4-4 Overview: Setting `max_connections`

In this practice, you review the response from the MySQL server when an attempt is made to open more connections than the setting of the `max_connections` system variable. The practice covers the following topics:

- Viewing the current `max_connections` system variable setting
- Executing a test against the MySQL server that produces connections greater than the `max_connections` system variable setting
- Increasing the `max_connections` system variable value and reattempting the test issued

ORACLE

Other Connection Status Variables

- `Connections`
  - This status variable displays the number of new connections established.
  - Connection pooling may be necessary if the value is too high.
- `Max_used_connections`
  - This status variable displays the maximum number of simultaneous connections.
  - If the value is equal (or close) to the `max_connections` setting, `max_connections` should be increased.
    - Possible sign of overload

Other Connection Status Variables

`Connections`

- If the `Connections` number is excessively high, you should consider using connection pooling. Numerous open source database connections pools are available (http://java-source.net/open-source/connection-pools).
- Connection pooling can provide improvement in applications where the queries executed are few and simple. Little improvement is gained if there are numerous connections executing complex queries.

Practice 4-5 Overview: Evaluating the Effect of Numerous Connections

In this practice, you study the effect that numerous connections have on the memory usage of the MySQL server by:

- Setting the `max_connections` system variable to be able to handle 1,024 connections
- Executing 100,000 requests against the MySQL server by using a progressive number of connections
- Evaluating the amount of virtual (swapped and physical) and physical (nonswapped) memory used by each execution

ORACLE

Other Thread Status Variables

- `Threads_cached`
  - This status variable displays the number of threads currently in the cache.
- `Threads_connected`
  - This status variable displays the current number of connections.
- `Threads_running`
  - This status variable displays the number of queries currently executing.
  - It provides a valuable picture of the load on your server.
  - View a running execution of the load on your server:

```
mysqladmin -i1 extended | grep Threads_running
```


Other Thread Status Variables

`Threads_cached`

- Value can be obtained by using `SHOW STATUS LIKE 'Threads_cached'`.

`Threads_running`

- This variable encompasses a large number of activities, such as queries waiting for IO, queries using locks, queries waiting to acquire a mutex, and so on.
- If there is no activity on the server, this is displayed as `1` (due to the `SHOW STATUS` statement being executed by the `mysqladmin` command).
  - Subtract 1 from the number to get the actual count.

Setting `thread_cache_size`: Scenario

The machine that houses your MySQL server is running at 90% to 95% of CPU processing. Issuing the following statement gives you a picture of the potential problem:

```
SHOW STATUS LIKE 'Threads%';
 +-------------------+--------+
 | Variable_name     | Value  |
 +-------------------+--------+
 | Threads_cached    | 0      |
 | Threads_connected | 54     |
 | Threads_created   | 345678 |
 | Threads_running   | 41     |
 +-------------------+--------+
```

What is your next step?

Setting `thread_cache_size`: Scenario

- There may be other status variables that you would want to look at such as `Uptime`, `Uptime_since_flush_status`, and `Max_used_connections`. If those values are reasonable, increasing the value of the `thread_cache_size` setting to a larger number has the potential to improve performance.

Per-Session Thread Buffers

- `sort_buffer_size`
  - Queries that use `ORDER BY` or `GROUP BY` options obtain a buffer size equivalent to this value.
- `read_rnd_buffer_size`
  - Queries that use `ORDER BY` or `GROUP BY` options use this buffer to read the rows necessary to output.
- `join_buffer_size`
  - Queries that execute joins without using indexes set the minimum amount of memory assigned by this setting to the query execution.
- `read_buffer_size`
  - Queries that execute full table scans use the size of this buffer to cache the row data.

Per-Session Thread Buffers

`sort_buffer_size`

- The default is 2 MB and can be considered large for OLTP applications.
  - Change this system variable to 128 KB and increase as necessary.
- When a query exceeds the available memory assigned in this buffer, a disk-based sort (disk seek) is used.

`read_rnd_buffer_size`

- The default is 256 KB and should be kept at this setting for most servers.
- Experiment with changing this system variable value only if you see a lot of rows being sorted.

`join_buffer_size`

- The default is 128 KB. Any increase in the size of this buffer can cause negative system performance.
  - Queries that use this setting uses the entire buffer size even if they do not require it.

read_buffer_size

- The default is 128 KB and should be set in increments of 4 KB.
- When queries perform writes, either through a `SELECT INTO ... OUTFILE` query or when merged results (when `filesort` is used) are written to a temporary file, this setting is used for buffering the output.
- There is no fixed formula for setting this variable. But with all variables, benchmarking different values is essential to obtaining the best value that should be used.

Per-Session Thread Buffers

- tmp_table_size
  - Determines the size of internal memory tables that can be stored
  - Associated status variables:
    - Created_tmp_disk_tables
    - Created_tmp_tables
    - Max_heap_table_size
- thread_stack
  - Determines the size of the stack assigned to each thread

Per-Session Thread Buffers (continued)

tmp_table_size

- When the MySQL server creates temporary tables to handle a query execution, the size of the internal memory tables that can be stored is determined by this setting.
- If the in-memory temporary table exceeds the available memory assigned by this buffer, an on-disk MyISAM table is created.
  - Excludes those tables that you create with `CREATE TABLE ... ENGINE=MEMORY`.
- When setting the value of this variable, the following status variables can be of assistance:
  - `Created_tmp_disk_tables` displays the number of on-disk temporary tables that were created to handle query executions.
  - `Created_tmp_tables` displays the total number of temporary tables that were created to handle query executions.

Note: You can compare the number of internal on-disk temporary tables created to the total number of internal temporary tables created by comparing the values of the `Created_tmp_disk_tables` and `Created_tmp_tables` variables. You should consider that each invocation of the `SHOW STATUS` statement uses an internal temporary table and increments the global `Created_tmp_tables` value.

`thread_stack`

- The default is dependent on the O/S and the number of bits supported. Reducing the size of this system variable can have the following effects:
  - The complexity of SQL statements that the server can handle is limited (could result in a thread stack overrun error).
  - Recursion depth of stored procedures along with other memory-consuming actions is also limited.

Quiz

When the MySQL server creates temporary tables to handle query execution, the size of the internal memory tables that can be stored is determined by which server setting?

a. `join_buffer_size`
b. `read_buffer`
c. `thread_stack`
d. `tmp_table_size`

Answer: d

Total Thread Memory Usage

- Total thread memory usage formula:
  - `max_connections` multiplied by per-thread buffers
  - Per-thread buffers:
    - `sort_buffer_size`
    - `read_rnd_buffer_size`
    - `join_buffer_size`
    - `read_buffer_size`
    - `thread_stack`
    - `tmp_table_size` or `max_heap_table_size` (smaller of the two)
- Global setting for these variables should be set low.
  - Increase these settings per session based on the needs of the client and the queries that are executed.

ORACLE

Total Thread Memory Usage

- The following formula determines the maximum estimate for thread memory usage:
  - `max_connections` x (`sort_buffer_size` + `read_rnd_buffer_size` + `join_buffer_size` + `read_buffer_size` + `thread_stack` + (`tmp_table_size` or `max_heap_table_size`))
- Not all these buffers are used at the same time. As a result, realistic thread memory use can be calculated as maximum thread memory usage divided by 4 (or 25% of the total):
  - `max_connections` x (`sort_buffer_size` + `read_rnd_buffer_size` + `join_buffer_size` + `read_buffer_size` + `thread_stack` + (`tmp_table_size` or `max_heap_table_size`)) / 4
- These formulas need to be adjusted based on observing your server's usage and must also take into account BLOBs, memory tables, closing of prepared statements, and other issues that have an effect on total memory usage.
  - Section 7.11.5.1 of the MySQL 5.5 manual discusses in greater detail how MySQL uses memory (http://dev.mysql.com/doc/refman/5.5/en/memory-use.html).
- Due to the potential for each thread to use a large portion of memory, the global settings for these variables should be low.

Total Thread Memory Usage: Scenario

On a server with 4 GB of memory, what is the potential problem with the following settings?

```
join_buffer_size = 128M
read_rnd_buffer_size = 256K
read_buffer_size = 128M
sort_buffer_size = 128M
thread_stack = 192K
max_connections = 151
max_heap_table_size = 16M
tmp_table_size = 16M
```


Total Thread Memory Usage: Scenario

- Using the realistic total thread memory usage formula could result in the following:
  - 151 x (128 MB + 256 KB + 128 MB +128 MB + 192 KB + 16 MB) / 4 would result in a possible estimated memory consumption of 15.11 GB.
- When setting these system variables, use low values for global settings and larger values for local settings when necessary.
  - These system variables can be set per-session for queries that benefit from large values.

Practice 4-6 Overview: Total Thread Memory Usage

In this practice, you calculate the total thread memory usage for the MySQL server by:

- Recording settings located in the `/etc/my.cnf` file
- Recording system variables from the `mysql` client
- Calculating the total thread memory usage by using the values collected

ORACLE

Answer: c

Optimal Sorting of Data

- Sort_merge_passes
  - This status variable displays the number of passes that were made during file merge sort.
    - If this number is large, consider increasing sort_buffer_size.
- Sort_range
  - This status variable displays the number of sorts that were completed using ranges.

Optimal Sorting of Data

Sort_merge_passes

- Verify that a file sort needs to be done at all.
  - Scenario: Using the employees table from the employees database, which SELECT statement would perform a file sort and which would not?

```
SELECT * FROM employees ORDER BY emp_no DESC LIMIT 1;
SELECT * FROM employees ORDER BY last_name DESC LIMIT
1;
```

Sort_range

- Example:

```
SELECT * FROM employees WHERE emp_no > 1000 AND emp_no
< 20000 ORDER BY last_name DESC LIMIT 1;
```

Optimal Sorting of Data

- `Sort_rows`
  - This status variable displays the actual number of rows that were sorted
    - Value provides a clue about how many complex sorts are being executed.
- `Sort_scan`
  - This status variable displays the number of sorts executed by scanning the full table.
    - Typical for sorts that are not using an indexed field

Practice 4-7 Overview: Sort Queries

In this practice, you determine the most effective setting for the `sort_buffer_size` system variable based on a sample of data collected. The practice covers the following:

- Setting the `sort_buffer_size` system variable by using settings ranging from 32k to 32m
- Executing a sample collection of SQL data using 20 connections against the MySQL server
- Evaluating the average time that the connections to the server take to complete, as well as the number of sort merge passes that the server must perform along with the amount of virtual (swapped and physical) and physical (nonswapped) memory used by each execution

ORACLE

Answer: c

Network Problems

- `Aborted_clients`
  - This status variable displays the number of connections that were terminated because the client died without closing the connection.
- `Aborted_connects`
  - This status variable displays the number of failed attempts to connect to your server.
- `Bytes_received`
  - This status variable displays the number of bytes received from all clients.
- `Bytes_sent`
  - This status variable displays the number of bytes sent to all clients.

ORACLE

Network Problems

`Aborted_clients`

- A large number here could be a sign of a network problem.
  - Verify that the `max_allowed_packet` setting is a reasonable size for the data types in your database.
  - Increasing `wait_timeout` can also help minimize this problem.

`Aborted_connects`

- This value should be zero.
  - Could be a sign of a network problem
  - Attacks on the server (bad password, port scans, and so on)
  - Coding issues, especially with PHP and the change in the default for the `default_socket_timeout` setting

`Bytes_received` and **`Bytes_sent`**

- Number of bytes sent to all clients; number of bytes received from all clients
- Can the network handle the amount of traffic?
- Are the numbers expected?

Quiz

Which status variable displays the number of failed attempts to connect to your server?

a. Aborted_clients
b. Aborted_connects
c. Bytes_received
d. Bytes_sent

Answer: b

Binary Logs Performance

- `binlog_cache_size`
  - This system variable defines the size of the cache to hold the SQL statements for the binary log during a transaction.
- `Binlog_cache_use`
  - This status variable displays the number of transactions that used the binary log cache.
- `Binlog_cache_disk_use`
  - This status variable displays the number of transactions that used the temporary binary log cache but exceeded the `binlog_cache_size` memory.

Binary Logs Performance

`binlog_cache_size`

- If your server supports transactions and the binary log is enabled, each client that connects has a binary log cache allocated.
- Increasing the size of the binary log cache can improve performance on systems that often execute large, multistatement transactions.

`Binlog_cache_use`

- Identifies the number of transactions that were able to fit in the `binlog_cache_size` memory

`Binlog_cache_disk_use`

- Excess statements that cannot be held in memory have to be stored on disk in a temporary file.
- Increasing the size of the `binlog_cache_size` setting minimizes the need to write and read from disk, which can improve performance.
- If the formula `Binlog_cache_disk_use` / `Binlog_cache_use` results in a large number, you should increase the size of `binlog_cache_size` to improve performance.

Quiz

Increasing the size of the `binlog_cache_disk_use` server setting minimizes the need to write and read from disk, which can improve performance.

a. True
b. False

Answer: b

All Statements

- `Com_*`
  - Numerous status variables that display the number of times each statement (`DELETE`, `INSERT`, etc.) has been executed
- `Queries`
  - This status variable displays the number of statements executed by the server from clients.
    - Includes those executed within stored programs
- `Questions`
  - This status variable displays the number of statements executed by the server from clients.
- `Slow_queries`
  - This status variable displays the number of queries that took longer than the time allotted in the `long_query_time` setting.

ORACLE

All Statements

`Com_*`

- Provides valuable information on the types of queries being executed on your server
- There is one status variable for each type of statement that can be executed.
- Are all the statements being executed expected?
  - Example: Should there be a value in the `Com_rollback` variable?

`Queries`

- Counts queries that were submitted to the server from the client, regardless of whether the server could execute them or not

`Questions`

- Does not count statements executed within stored programs
- Similar to `Queries` status variable: Counts all statements submitted from the client, including malformed queries
- Provides a rough load indicator or stable load mix

`Slow_queries`

- These queries are considered to be slow and are logged in the slow query log if it is enabled.
  - You should enable the slow query log it is not enabled.
- Monitor this setting to determine if the queries executed against your server are optimally tuned.

Quiz

Which status variable displays the number of statements executed by the server from clients, including those executed within stored programs?

a. Com_rollback
b. Queries
c. Questions
d. Slow_queries

Answer: b

SELECT Statements

- Select_full_join
  - This status variable displays the number of joins that did not use indexes.
- Select_full_range_join
  - This status variable displays the number of joins that used a range search on a referenced table.
- Select_range
  - This status variable displays the number of joins that used ranges on the first table.

SELECT Statements

Select_full_join

- For optimal server performance, this value should be 0.
- Locate and correct join SQL statements that do not use indexes.

Select_full_range_join

- These are rare. But if they appear, they may have a negative effect on performance.
- Locate and determine if they can be optimized to improve performance.

Select_range

- If this status variable is a large number, research the cause and optimize.

Select_range_check

- These queries take a lot of overhead.
- Optimize these queries to improve performance.

SELECT Statements

- Select_range_check
  - This status variable displays the number of joins with keys that check for key usage after each row.
- Select_scan
  - This status variable displays the number of joins that performed a full table scan on the first table.

SELECT Statements

Select_scan

- If this status variable is a large number, research the cause and optimize.
- Setting log_queries_not_using_index and setting long_query_time to 600 will write the majority of queries performing full table scans to the slow query log. This can be an excellent tool to locate queries that perform full table scans.

Quiz

Which status variable displays the number of joins that used a range search on a table referenced?

a. `Select_full_join`
b. `Select_full_range_join`
c. `Select_range`
d. `Select_range_check`

Answer: b

Table Locks Performance

- `Table_locks_immediate`
  - Displays the number of times that a table lock was granted immediately
- `Table_locks_waited`
  - Displays the number of times that a table lock was not granted immediately
  - Possible solutions:
    - Convert to InnoDB tables.
    - Partition tables.
    - Optimize the queries.
    - Use concurrent inserts.
    - Fix lock settings.

ORACLE

Table Locks Performance

`Table_locks_immediate`

- Any delays in obtaining table locks will negatively affect performance.
- For InnoDB tables, waits are rare.

`Table_locks_waited`

- Any delay in obtaining a lock is a performance issue.
  - There is no specific information about how long the wait takes.
- A large value for this status variable points to a serious performance bottleneck.

Quiz

For InnoDB tables, table lock waits are rare.

a. True
b. False

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe the advantages and limitations of the MySQL server architecture that directly affect performance
- Use server settings and status variables to evaluate and alter the performance of the MySQL server

5

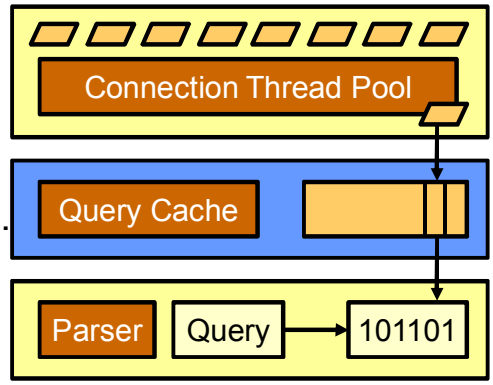
MySQL Query Cache

Objectives

After completing this lesson, you should be able to:

- Describe the advantages and limitations of the MySQL query cache in relation to server performance
- Edit the query cache server settings
- Explain the query cache status variables
- Improve query cache results

MySQL Query Cache



- Caches the full result set produced from a `SELECT` query:
  - Queries executed must be identical.
  - Cache is stored in system memory.
  - Cache is checked before the query is parsed.
- Associated table updates invalidate query cache results.
- Prepared statements can be cached, but limitations exist.
- Some statements that do not use the query cache:
  - Queries that use nondeterministic functions
  - Queries that are a subquery of an outer query
  - Queries that are executed within the body of a stored function, trigger, or event

ORACLE

MySQL Query Cache

Disabling Query Cache

The query cache should be disabled in a server with many fast queries and many CPU cores. The query cache runs in a single thread and can severely limit throughput in a server.

MySQL Query Cache Settings

- query_cache_type
  - Sets the type of query caching
    - 0 or OFF: Turns off the query cache
    - 1 or ON: Caches all cacheable queries
    - 2 or DEMAND: Caches only SELECT SQL_CACHE queries
  - To completely turn off, use --query-cache-type=0.
- query_cache_size
  - Sets the amount of memory allocated for caching query results
- query_cache_limit
  - Sets the maximum result set size that can be cached

MySQL Query Cache Settings

query_cache_type

- Three individual options can be set:
  - **0 or OFF:** This stops the query cache by not caching query results or retrieving results from the query cache.

    Note: This option does not deallocate the query cache buffer but does prevent the fast taking and releasing of the query cache mutex.
  - **1 or ON:** Caches all cacheable query results except for those that begin with SELECT SQL_NO_CACHE ...

    This is the default option for backward compatibility.
  - **2 or DEMAND:** Caches results only for cacheable queries that begin with SELECT SQL_CACHE ...
- Starting MySQL server with --query-cache-type=0 does not acquire the query cache mutex, which reduces the overhead during query execution.
  - Up to 13% improvement is possible.

query_cache_size

- Amount of memory allocated for caching query results:
  - The default value is `0`, which disables the query cache.
  - Setting this value too low (anything under approximately 40 KB) results in a warning being issued.
  - Setting this value too large can result in all queries blocked while invalidating results from changed tables.
  - All values should be entered as multiples of 1024. Any other values are rounded down to the nearest multiple.
  - On most production servers, 8 MB is fine. But 128 MB to 200 MB can also be acceptable if there is ample evidence to support using such a large value.

query_cache_limit

- Maximum result set size to cache. The default is 1 MB.
- Keeping this at a reasonable number (1 MB to 2 MB) avoids erasing the entire query cache due to a large query being stored.
  - Actual size depends on application needs and `SELECT` result set sizes.

When *Not* to Use the MySQL Query Cache

- When the database manages numerous fast queries and the server running the database uses multiple CPU cores
- When your database has multiple writes, updates, or deletes to tables taking place
  - Every time a table is modified, the queries and results associated with those tables in the query cache are discarded.
- In situations where there are numerous different `SELECT` queries being executed and stored in the query cache
- In situations where you lock tables to prevent users from seeing any table data

ORACLE

When *Not* to Use the MySQL Query Cache

- Numerous fast queries and multiple CPU cores
  - Most servers running MySQL use multiple CPU cores. Using the MySQL query cache in these servers would severely degrade performance.
- Multiple writes, updates, or deletes
  - The overhead associated with the query cache will degrade performance.
- Numerous different `SELECT` queries
  - Every result set needs to be indexed in the query cache. If more complete result sets are being stored in the query cache, performance gains will be smaller.
- Locked tables
  - The query cache is checked before the query is parsed, which can result in a previous result set being outputted before the server ever checks to see if the table is locked.
  - When the `query_cache_wlock_invalidate` system variable is turned on, the query cache is invalidated if any object that it uses has a write lock executed against it.

When to Use the MySQL Query Cache

- When the database must repeatedly run the same queries against the same data set
  - Saves the MySQL server from having to retrieve the same data set over and over again
- When the result set is relatively small and manageable by the query cache
  - Having large result sets in the query cache can have a negative effect on performance.
- Typical uses of query cache in production:
  - Reading intensive web applications.
    - Blogs, eCommerce applications, advertising applications, and so on

Quiz

Which of the following is *not* a scenario in which you should use the MySQL query cache?

a. In web applications that primarily are read-intensive (blogs, advertising applications, and so on)
b. When your database has multiple writes, updates, or deletes to tables taking place
c. When the database must repeatedly run the same queries against the same data set
d. When the result set is relatively small and manageable by the query cache

Answer: b

MySQL Query Cache Status Variables

- `Qcache_hits`
  - Displays the number of times that results were delivered from the query cache
- `Qcache_inserts`
  - Displays the number of times that results were stored to the query cache
    - If the number of query cache hits is low and the number of query cache inserts is high, you should disable the query cache.
- `Qcache_lowmem_prunes`
  - Displays the number of stored result sets that were deleted from the query cache due to low memory
    - If this number is high, you can increase the query cache size.

MySQL Query Cache Status Variables

- `Qcache_free_blocks`
  - Displays the number of free contiguous blocks of memory in the query cache
- `Qcache_free_memory`
  - Displays the amount of free memory for the query cache
    - If free memory is small (in relation to the query cache size), you should increase the query cache size.
- `Qcache_not_cached`
  - Displays the number of `SELECT` queries that did not have their result sets stored in the query cache.

ORACLE

MySQL Query Cache Status Variables

`Qcache_free_blocks`

- The allocation time for new queries is proportional to the value that is entered in this system variable.

`Qcache_not_cached`

- These queries could have been left out of the query cache because they were not cacheable or based on the setting of the `query_cache_type` setting.

MySQL Query Cache Status Variables

- `Qcache_queries_in_cache`
  - Displays the number of query result sets that are currently stored in the cache
- `Com_select`
  - Displays the number of `SELECT` statements executed by the MySQL server
    - This does not include queries that were served from the query cache.
  - Compare the `Com_select` and `Qcache_hits` status variables.

MySQL Query Cache Status Variables (continued)

`Com_select`

- Comparing `Com_select` and `Qcache_hits` indicates the efficiency of the query cache on your server.
- Add the `Com_select` and `Qcache_hits` status values together to obtain the number of `SELECT` statements executed against the server.

Practice 5-1 Overview: Query Cache

In this practice, you determine the most effective setting for `query_cache_size` server variable based on a sample of data collected. The practice covers the following topics:

- Setting the `query_cache_size` system variable by using settings ranging from 0 to 16m
- Executing a sample collection of SQL data by using 10 connections against the MySQL server
- Evaluating the average time that the connections to the server take to complete along with the amount of virtual (swapped and physical) and physical (nonswapped) memory used by each execution
- Calculating the query cache utilization rate for each setting

ORACLE

Quiz

Comparing `Com_select` and `Qcache_inserts` indicates the efficiency of the query cache on your server.

a. True
b. False

Answer: b

Improve Query Cache Results

- Standardize the queries that are executed against the server.
  - If possible, remove anything that prevents the query cache from being able to store the result set.
- Minimize the number and frequency of updates to the data.
  - Batch updates together to minimize the query cache result sets from being invalidated on a frequent basis.

Quiz

Batch updates together to minimize the query cache result sets from being invalidated on a frequent basis.

a. True
b. False

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe the advantages and limitations of the MySQL query cache in relation to server performance
- Edit the query cache server settings
- Explain the query cache status variables
- Improve query cache results

6 InnoDB

Objectives

After completing this lesson, you should be able to:

- Explain the best uses for the InnoDB storage engine
- Describe the InnoDB log files and buffers
- Design tables to take advantage of using the InnoDB storage engine
- Explain the `SHOW ENGINE INNODB STATUS` output
- Set up the InnoDB monitor
- Edit the InnoDB server settings

ORACLE

InnoDB Storage Engine

- Default storage engine
- Supports full ACID compliance
- Supports `COMMIT`, `SAVEPOINT`, and `ROLLBACK`
- Enforces referential integrity through foreign key constraints
- Provides auto-recovery after crash
- Supports multiversioning (MVCC) and row-level locking
- Clustered by primary key (implicit if not defined)

ORACLE

InnoDB Storage Engine

Default storage engine

- Reliability
- Concurrency
- Good performance on modern systems

ACID

- Atomic, consistent, isolated, and durable
- InnoDB supports a wide range of isolation modes from `READ-UNCOMMITTED` to `SERIALIZABLE`.

MVCC

- True multiversion concurrency control (MVCC) enables records and tables to be updated without the overhead associated with row-level locking mechanisms.
- The MVCC implementation in InnoDB virtually eliminates the need to lock tables or rows during the update process.

InnoDB Storage Engine: Uses

- Critical data applications
- Creating transactional applications
- Creating heavy concurrency applications
- Minimizing down time after server crash
- Increasing speed of access using the primary key
  - Improves memory performance
  - Primary key joins are fast.
- Storing binary large objects (BLOBs) in the same table as primary data

ORACLE

InnoDB Storage Engine: Uses

Transactional applications

- Applications that have lots of small writes
- Applications with related changes across multiple tables

Primary key

- Additional indexes reference the row by means of the primary index.
- Keeping the primary key short improves performance.
- Primary key lookups are more efficient than secondary key lookups.
- Joins on primary keys provide the best performance.

BLOBs

- InnoDB ignores BLOBs if they are not referenced in the `SELECT` statement.
- BLOBs can be kept in same table with little (or no) performance degradation.
  - COMPACT row format is best.
- BLOBs are allocated to new space on update.

Using the InnoDB Storage Engine

- Depending on the workload, transaction overhead can affect performance.
- InnoDB is suited for systems with larger disk space and higher memory.
- Key compression is not supported.
- Accurate row counts require additional work.
- Accurate InnoDB statistics require additional work.
  - Increase the size of the `innodb_stats_sample_pages` server variable to improve statistics.

Using the InnoDB Storage Engine

Transaction overhead

- The advantage of transactional storage engines outweighs any minor performance degradation for most applications.

InnoDB systems

- More memory is necessary to perform updates.

Compressed InnoDB indexes

- InnoDB indexes are compressed only when the `ROW_FORMAT` option is set to `COMPRESSED`.

Counts without `WHERE`

- Counts with `WHERE` clauses produce better performance.
  - `SELECT COUNT(`*`indexed_column`*`)` produces better results than a nonindexed column.
- InnoDB performs a table/index scan to produce the result. This is instant for MyISAM tables.

- Other means to obtain row count can be inaccurate:
  - `SHOW TABLE STATUS LIKE 'table'` produces an approximate number of rows.
- InnoDB uses estimation for each query execution:
  - Uses random BTREE dives
- Creating a counter table can improve performance. The disadvantage is the need to keep the table up to date.

InnoDB statistics

- Cardinality values are produced by using a BTREE dive.
- Statistics are produced the first time a table is opened after server startup.
  - This causes a slight delay the first time the table is opened.
- Performing `ANALYZE TABLE` produces more accurate results over time.

  Note: For more information about controlling the optimizer's statistical estimations, see http://dev.mysql.com/doc/innodb-plugin/1.1/en/innodb-other-changes-statistics-estimation.html.

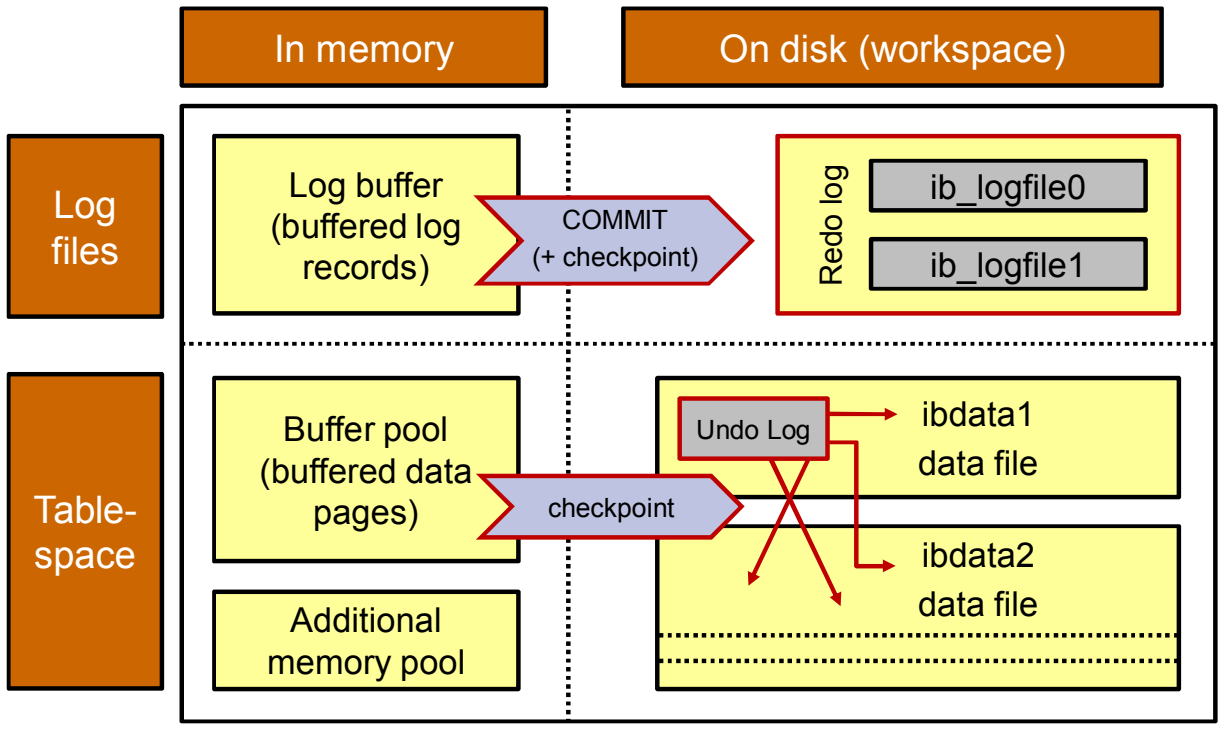
Quiz

Which of the following is *not* a good use for the InnoDB storage engine?

a. Applications that require bulk data loads
b. Storing BLOBs in the same table as primary data
c. Creating transactional applications
d. Creating heavy concurrency applications

Answer: a

InnoDB Log Files and Buffers



ORACLE

InnoDB Log Files and Buffers

InnoDB Log

- As a client performs a transaction, the changes that it makes are held in the InnoDB log.
  - The more recent log contents are cached in memory.
  - The cached log information is written and flushed to log files on disk at transaction commit time.
    - This can also occur earlier, depending on the length of the transaction.
- If a crash occurs while the tables are being modified, the log files are used for auto-recovery.
  - When the MySQL server restarts, the changes recorded in the logs are reapplied.
    - This ensures that the tables reflect all committed transactions.

Quiz

As a client performs a transaction, the changes that it makes are held in which log?

a. Checkpoint log
b. InnoDB log
c. Slow query log
d. Transaction log

Answer: b

Practice 6-1 Overview: Commit Transactions

In this practice, you evaluate the effect of the frequency of commits on performance by:

- Evaluating a MySQL stored procedure called `innodb_inserts`
- Using the `many_tables` database to execute multiple instances of the `innodb_inserts` MySQL stored procedure

ORACLE

InnoDB Table Design

- Use a short, integer primary key.
  - Primary key is used with all indexes created.
    - A copy of the primary key is stored in each index.
  - `INT` keys are more efficient than `VARCHAR/CHAR` keys.
- Create an artificial primary key.
  - Add `PRIMARY KEY AUTO_INCREMENT` column and change current `PRIMARY KEY` to `UNIQUE`.
- Look for ways to create and use covering indexes.
- If you use composite indexes or composite primary keys, the order of the columns can affect performance.
- Consider using BLOBs for long rows.
  - The `BLOB` column is only read if it is part of the `SELECT` statement; otherwise, the column is ignored.

ORACLE

InnoDB Table Design

InnoDB primary key

- Choose primary key columns that are stable and never change, because changing a primary key value is an expensive operation.
- If the primary key column is a long string, use a prefix index so that the keys include only enough text to distinguish and sort the values.
- Use secondary indexes for important queries that do not reference the primary key.
  - Create a secondary index on the most “selective” columns for important queries.
- InnoDB creates a clustered key internally if no primary key exists in your table. It is more efficient for you to create a primary key and use it rather than have InnoDB create an internal key.

`UNIQUE` indexes

- If a column really is unique, declaring a unique index has advantages over a nonunique one.
  - Example: `SELECT COUNT(*)` is faster for that table.

Covering indexes

- If the `SELECT` list and `WHERE` clauses reference only the columns from <secondary index + primary key columns>, the query can get all the data from the secondary index and does not need to look up and read the full row using the primary key.

Composite index and composite primary key

- These types of indexes are multicolumn indexes that can be used for queries involving different combinations of columns.

Quiz

Which of the following is a true statement about InnoDB table primary keys?

a. Choose primary key columns that are stable and never change, because changing a primary key value is an expensive operation.
b. Create secondary indexes on every column so that the primary key does not need to be used.
c. It is more efficient to have InnoDB create an internal key than for you to create a primary key.
d. Primary keys should not be prefixed, regardless of the length of the string.

ORACLE

Answer: a

SHOW ENGINE INNODB STATUS

- Transactions
  - Displays active and pending transactions:
    - MySQL thread ID, IP address, MySQL username, and the task that MySQL is performing
- Buffer pool and memory
  - Displays buffer pool activity and memory usage:
    - Total memory allocated by InnoDB, amount of memory allocated in additional pool memory, total number of pages in buffer pool, number of pages free, pages allocated by database pages, and dirty pages
- Row operations
  - Displays current and historical query activity against the MySQL server

ORACLE

SHOW ENGINE INNODB STATUS

Transactions

- `Purge done for trx's n:0` displays the number of purged transactions.
  - Uncommitted transactions may become stale and block the purge process.
- `undo n:0` displays the undo log record number for the purge that is currently processing.
- `History list length` displays the number of transactions that have not been purged from undo space.
- Status of connection:
  - `not started`: No active InnoDB transaction is taking place with this connection.
  - `ACTIVE`: An active InnoDB transaction is taking place with this connection.
  - `sleep`: The transaction is delayed until a free slot is available in the InnoDB queue.
    - Adjusting the `innodb_thread_sleep_delay` setting may be necessary.
- Status of transaction can include “fetching rows,” “updating,” and other self-explanatory identifiers.
  - If transactions are waiting on locks, there is a possible deadlock situation.

Buffer pool and memory

- Provide valuable insight into how well your buffer pool is sized:
  - If there are numerous pages free, you may need to reduce your buffer pool size (`innodb_buffer_pool_size`).
- Buffer pool hit rate is system and workload dependent:
  - `1000/1000` identifies a 100% hit rate.
  - Slightly lower hit rate values may be acceptable.

Row operations

- InnoDB thread queue status:
  - Displays the number of threads that are inside InnoDB and active
  - Displays the number of threads that are currently waiting in the InnoDB queue
- `Read views open inside InnoDB` displays the number of transactions that were started but are not currently active.
- The state of the main InnoDB thread:
  - Controls the scheduling of a number of system operations
  - The values are self-explanatory.
- Number of rows operations affected by `INSERT`, `UPDATE`, `DELETE`, and `SELECT` statements on InnoDB tables.
  - These values are counted from system startup.
  - Also displayed is the average number of row operations for each transaction.

Note: `SHOW ENGINE INNODB STATUS` can be misleading at times because the values displayed are collected at different times. The overhead required to obtain a global lock to provide consistent information throughout the report would itself produce a performance drag on the system. Comparing the results over a period of time provides more accurate and useful information to help you improve system performance.

Other `SHOW ENGINE INNODB STATUS` Sections

- Semaphores
- Latest detected deadlock
- Latest foreign key error
- File I/O
- Insert buffer, change buffer, and adaptive hash index
- Log

Other `SHOW ENGINE INNODB STATUS` Sections

Semaphores

- Displays the various locks used inside InnoDB
- Values that are high indicate a busy server with frequent contention inside InnoDB.
- If your server has been running longer, the values become higher because the values are cumulative.

Latest detected deadlock

- Displays the last transactions that caused deadlocks
- Includes the state of the transaction during deadlock, the locks they were holding and what they were waiting for, and which transactions were rolled back to resolve the deadlock
- For complex deadlock issues, it is better to investigate the log files.

Latest foreign key error

- Displays the last transactions that caused a foreign key error
- Includes the definition of the foreign key that failed, along with what InnoDB considers the closest match in the parent table

InnoDB Support in INFORMATION_SCHEMA

- INNODB_CMP and INNODB_CMP_RESET
  - Contains status information about the operations related to compressed tables
- INNODB_CMPMEM and INNODB_CMPMEM_RESET
  - Contains status information about the compressed pages that reside in the buffer pool
- INNODB_TRX
  - Contains information about every transaction currently executing inside InnoDB
- INNODB_LOCKS and INNODB_LOCK_WAITS
  - Contains information about the locks that the InnoDB storage engine is currently holding

InnoDB Support in INFORMATION_SCHEMA

INNODB_CMP and **INNODB_CMP_RESET**

- These two tables have identical contents, but reading from the INNODB_CMP_RESET table resets the statistics on compression and uncompression operations.

INNODB_CMPMEM and **INNODB_CMPMEM_RESET**

- These two tables have identical contents, but reading from the INNODB_CMPMEM_RESET table resets the statistics on relocation operations.

INNODB_TRX

- Includes whether the transaction is waiting for a lock, when the transaction started, and which SQL statement the transaction is executing

INNODB_LOCKS

- Contains one row for each lock that is blocking another transaction
- Includes the state of the transaction that holds the lock (RUNNING, LOCK WAIT, ROLLING BACK, or COMMITTING)

INNODB_LOCKS_WAITS

- This table contains one or more rows for each blocked transaction, indicating the lock that it has requested and any locks that are blocking that request.

Practice 6-2 Overview:
SHOW ENGINE INNODB STATUS

In this practice, you evaluate the output of the `SHOW ENGINE INNODB STATUS` command by:

- Using the `many_tables` database to create a large table that is used with this practice
- Executing the `mysqlslap` application running multiple iterations of the `innodb_query.sql` script by using multiple concurrent connections
- Executing and answering questions related to the `SHOW ENGINE INNODB STATUS` output

ORACLE

Quiz

Which section of the `SHOW ENGINE INNODB STATUS` report displays the state of the main InnoDB thread?

a. Buffer pool and memory
b. File I/O
c. Log
d. Row operations

Answer: d

InnoDB Monitors

- Types:
  - Standard InnoDB monitor
    - Create a table called `innodb_monitor`.
  - Standard InnoDB monitor with additional lock information
    - Create a table called `innodb_lock_monitor`.
  - Standard InnoDB monitor with additional tablespace information
    - Create a table called `innodb_tablespace_monitor`.
  - Standard InnoDB monitor with additional internal data dictionary information
    - Create a table called `innodb_table_monitor`.
- Contents of created table are ignored.
  - `CREATE TABLE innodb_monitor (a INT) ENGINE=INNODB;`

InnoDB Monitors

Documentation

- For a complete discussion about InnoDB monitors, refer to the MySQL documentation at http://dev.mysql.com/doc/refman/5.5/en/innodb-monitors.html.

`PERFORMANCE_SCHEMA`

- In addition, the `PERFORMANCE_SCHEMA` database provides valuable information to monitor the InnoDB tables.
- More information can be found at http://dev.mysql.com/doc/refman/5.5/en/performance-schema.html.

Practice 6-3 Overview: InnoDB Monitors

In this practice, you evaluate the output of the available InnoDB monitors by:

- Making a backup copy of the `mysql` error log, deleting the error log, and restarting the server
- Creating a table that is the trigger to output the InnoDB data dictionary information about the tablespace to the `mysql` error log
- Making a backup copy of the mysql error log, deleting the error log, and restarting the server
- Creating a table that is the trigger to output the contents of the InnoDB internal data dictionary
- Executing and answering questions related to the table output

ORACLE

Quiz

Which of the following tables produces a report consisting of a standard InnoDB status report with additional internal data dictionary information?

a. `innodb_monitor`
b. `innodb_lock_monitor`
c. `innodb_table_monitor`
d. `innodb_tablespace_monitor`

Answer: c

InnoDB Settings

- `innodb_buffer_pool_size` sets the size (in bytes) of the memory buffer that InnoDB uses to cache data and indexes of its tables.
  - Start with 60% to 80% of memory on InnoDB-only systems.
  - This is especially important for write-intensive workloads.
- `innodb_log_file_size` sets the size (in bytes) of each log file in a log group.
  - Should be set as large as possible
- `innodb_log_files_in_group` sets the number of log files in the log group.
  - The default is `2` and should remain unchanged.

InnoDB Settings

`innodb_buffer_pool_size` server variable

- Default is 128 MB.
- Unlike MyISAM, which caches only indexes, InnoDB caches both data and indexes.

`innodb_log_file_size` server variable

- Preferred size is 2 GB so that adaptive flushing can work well.
  - For small buffer pools, try 50% of buffer pool size to start.
- Combined size of log files must be less than 4 GB.
  - This value is a product of `innodb_log_file_size` and `innodb_log_files_in_group`.
- The larger the value, the less checkpoint flush activity is needed in the buffer pool, thereby saving disk I/O.
- Use a command such as `mtime` to evaluate how frequently the log file changes.
  - This provides an indication of how effective the setting is.
- The setting should be big enough so that it can hold at least an hour of logs.

`innodb_log_files_in_group` server variable

- InnoDB writes to the files in a circular fashion.

InnoDB Settings

- `innodb_additional_mem_pool_size` sets the size (in bytes) of a memory pool that InnoDB uses to store data dictionary information and other internal data structures.
- `innodb_autoextend_increment` sets the increment size (in MB) for extending the size of an auto-extending shared tablespace file when it becomes full.
  - Set to1% to 5% of your file system disk space.
- `innodb_file_per_table` defines whether InnoDB tables are created in per-table tablespaces.
  - Is set to `OFF` (disabled) by default

InnoDB Settings (continued)

`innodb_additional_mem_pool_size` server variable

- Increased automatically and not worth setting
  - Estimated from InnoDB status to equal:
    Total memory allocated – buffer pool size * 16 * 1024
- It is recommended that you set the `use_sys_malloc` server variable to `1` to ignore this setting.

`innodb_autoextend_increment` server variable

- Default is 8 MB (not used for `innodb_file_per_table` tablespaces).
- Larger size helps the operating system allocate disk space in a nonfragmented way.

`innodb_file_per_table` server variable

- If this server variable is enabled, InnoDB creates each new table by using its own `.ibd` file for storing data and indexes, rather than the shared tablespace.
- For solid state drive (SSD) systems, store system tablespace on normal disk and store per-table tablespaces on SSD. This prevents the buffer and undo logs from wasting SSD write capacity with sequential writing.
- The advantages of turning this setting on include:
  - Support for compression and compact row features
  - Concurrent writes with `innodb_flush_method = O_DIRECT` even on ext* file systems
  - Ability to regain space per tablespace
  - Ability to monitor table size and date of last modification on the file-system level
- The disadvantage of turning this setting on is increased file-system fragmentation.
  - Defragmenting can sometimes help.
  - This applies to Linux file systems as well.
  - Copy the file somewhere else and then copy it back.

InnoDB Settings

- `innodb_flush_log_at_trx_commit` defines how often the log buffer is written out to the log file and when the flush to disk operation takes place.
  - `0` sets the tasks to perform once per second (fastest).
  - `1` sets the tasks to perform at transaction commit (slowest).
  - `2` sets the log buffer to be written out to disk once per second, but the flush to disk task is not performed (fast).
- `innodb_log_buffer_size` sets the size (in bytes) of the buffer that InnoDB uses to write to the log files on disk.
  - Set `sync_binlog` to `0` to allow concurrent transaction commits.
  - Set to a higher value if you are performing large transactions.

InnoDB Settings (continued)

`innodb_flush_log_at_trx_commit` server variable

- `1` is the default and required for InnoDB to be considered ACID compliant.
  - InnoDB's crash recovery works regardless of the setting.
- Setting this variable to `0` or `2` can improve performance, but at the cost of losing approximately one second of transactions in a crash.
  - Any `mysqld` process crash can erase approximately one second of transactions when this variable is set to `0`.
  - Only an operating crash or power outage can erase approximately one second of transactions when this variable is set to `2`.
- For more details about this server variable setting, see the MySQL documentation at http://dev.mysql.com/doc/refman/5.5/en/innodb-parameters.html#sysvar_innodb_flush_log_at_trx_commit.

InnoDB Settings

- `innodb_flush_method` defines the process to use to flush both the data and log files.
  - `fdatasync` uses `fsync()` and is the default setting.
    - On Windows systems, the only available option is `async_unbuffered`.
  - `O_DSYNC` uses `O_SYNC` to open and flush the log files and `fsync()` to flush the data files (usually slow).
  - `O_DIRECT` uses `O_DIRECT` to open and flush the log files and `fsync()` to flush the data files.
  - Linux

InnoDB Settings (continued)

`innodb_flush_method` server variable

- There have been problems with using `O_DSYNC` on many varieties of Unix systems.
- `O_DIRECT` is available on some GNU/Linux versions, FreeBSD, and Oracle Solaris.
  - `directio()` is used on Oracle Solaris systems.
- Test all the options.
  - Do not assume that `O_DIRECT` is always best.

InnoDB Settings

- `innodb_thread_concurrency` defines the number of operating system threads that can run concurrently in InnoDB:
  - `0` is interpreted as infinite concurrency (no concurrency checking).
  - If a limit is used, set `innodb_concurrency_tickets` initially to `5000` and adjust so that 99% of queries run with one allocation.

InnoDB Settings (continued)

`innodb_thread_concurrency` server variable

- Default is `0`.
- Range is `0` to `1000`.
  - `0` is best for low-load servers and high-load servers if the connection pool limits concurrent threads to the range of 16 to 32 (it is in this range where performance is fastest).
- Set a limit for this status variable if long-running queries show up and there is a big increase in connections.
- This server variable can be changed while the server is running.
  - Change the server variable setting, wait a minute, and then check the connection count. Repeat this process to find the best setting.

innodb_concurrency_tickets

- If a limit is set for the `innodb_thread_concurrency` server variable, it is important to set this server variable to reduce queue overhead.

InnoDB Settings

- `innodb_purge_threads` defines the number of background threads devoted to the InnoDB purge operation.
  - Set to `1` to allow the purge operations to keep up.
- `innodb_io_capacity` defines the maximum number of I/O operations per second that InnoDB will perform.
- `innodb_buffer_pool_instances` defines the number of regions into which the InnoDB buffer pool is divided.
  - Set to `8` when your system has 16 cores or more.

InnoDB Settings (continued)

`innodb_purge_threads` server variable

- Default is 0.
- Monitor the history list length (in the Transactions section of the `SHOW ENGINE INNODB STATUS` report).
- Set the `innodb_max_purge_lag` server variable to `10,000,000` as a further precaution.

`innodb_io_capacity`

- Increasing this server variable allows purge operations to keep up.
- On fast SSDs with a high write load, a setting of `5,000` may be needed.

`innodb_buffer_pool_instances`

- Improves throughput gain by as much as 10%
- It is possible to see as much as a 30% throughput gain with 32 cores.

InnoDB Setup: Example

- Single/Master server.
  - `innodb_flush_log_at_trx_commit = 1`
- Slave, server with no binlog, unsafe with binlog
  - `innodb_flush_log_at_trx_commit = 1`
    - Can be set to safer setting if desired
  - `innodb_support-xa = 0`
    - Unsafe with binlog on!
  - binlog off
- Both master and slave
  - `sync_binlog = 0`
  - `innodb_log_file_size = 2G`
  - `innodb_purge_threads = 1`

ORACLE

InnoDB Setup: Example

- Set `innodb_io_capacity`, `innodb_buffer_pool_instances`, and `innodb_thread_concurrency` appropriately.
- For Linux systems, use XFS or ZFS file systems (avoid ext*), set swappability, and set the I/O scheduler.
  - Consider setting `numactl –interleave all`.
  - See http://jcole.us/blog/archives/2010/09/28/mysql-swap-insanity-and-the-numa-architecture/.
- To reduce transaction overhead when migrating from MyISAM, set `innodb_flush_log_at_trx_commit` to `0`.

Practice 6-4 Overview: InnoDB Settings

In this practice, you evaluate the output of the additional InnoDB monitors by:

- Viewing the current InnoDB server settings
- Creating a load against the MySQL server by using the `mysqlslap` application
- Modifying the `innodb_flush_log_at_trx_commit` and `innodb_buffer_pool_size` server variables between runs and making note of any changes in performance
- Viewing the amount of virtual and resident memory that the MySQL server is consuming while the `mysqlslap` application is running

ORACLE

Quiz

If your application has more InnoDB tables, you need to allocate more memory to which of the following InnoDB server variables?

a. `innodb_additional_mem_pool_size`
b. `innodb_autoextend_inrement`
c. `innodb_buffer_pool_size`
d. `innodb_log_buffer_size`

Answer: a

Summary

In this lesson, you should have learned how to:

- Explain the best uses for the InnoDB storage engine
- Describe the InnoDB log files and buffers
- Design tables to take advantage of using the InnoDB storage engine
- Explain the `SHOW ENGINE INNODB STATUS` output
- Set up the InnoDB monitor
- Edit the InnoDB server settings

7 MyISAM

Objectives

After completing this lesson, you should be able to:

- Explain the best uses for the MyISAM storage engine
- Design tables to take advantage of using MyISAM
- Optimize MyISAM tables
- Describe MyISAM table locks
- Edit the MyISAM server settings
- Describe the MyISAM key cache
- Explain the MyISAM key cache status variables
- Use MyISAM full-text indexing capabilities

ORACLE

MyISAM Storage Engine

- MyISAM is nontransactional storage engine.
  - Corruption can occur on power down.
- MyISAM uses a small disk and memory footprint.
- MyISAM indexes
  - MyISAM can work without indexes.
- MyISAM uses table-level locking.
- Inserts to MyISAM tables can be concurrent inserts.
- When MyISAM is compressed with `myisampack`, data and indexes are read-only.
- Only MyISAM indexes are cached by the MySQL server.
  - MyISAM data is cached by the O/S.

ORACLE

MyISAM Storage Engine

MyISAM indexes

- `B-TREE`, `FULLTEXT`, and `RTREE` indexes are available with MyISAM tables.

Compressed MyISAM

- The `myisampack` command compresses MyISAM tables up to 70%.
  - This is accomplished by compressing each column in the table separately.
  - **Note:** For more information about the `myisampack` command, see http://dev.mysql.com/doc/refman/5.5/en/myisampack.html.

MyISAM Storage Engine: Uses

MyISAM is useful in the following situations:

- Applications that do not require transactional capabilities or have a low write/update/delete concurrency
  - Example: Logging applications
- Applications that are read-only (or read-mostly)
- When full table scans are necessary
  - Example: Creating reports
- Applications that require bulk data loads
- Applications that require data crunching

MyISAM Storage Engine: Uses

It is recommended that you use the InnoDB storage engine unless you are sure that the MyISAM storage engine is needed. The advantages of the InnoDB storage engine crash safety and its up-time benefit are usually more significant than the possible benefits of the MyISAM storage engine in some situations.

MyISAM Table Design

- Declare columns `NOT NULL`.
  - This saves space.
- Use fixed-width columns for entire table definition.
  - Usually faster due to less row fragmentation
  - Easier for indexes to reference the data file
    - Variable-width columns force the data file pointer to store values based on the byte offset.
    - Example: Instead of reference rows being 1,2,3, and so on, variable widths reference offsets that could be 1, 56, 272, and so on.
  - Easier to fix if corruption occurs
    - It is possible for variable-width pointers to become corrupted as well.

ORACLE

MyISAM Table Design

Static versus dynamic table characteristics

- Static table format is the default for MyISAM tables.
  - Used when the table contains no variable-length columns (`VARCHAR`, `VARBINARY`, `BLOB`, or `TEXT`)
  - Each row is stored using a fixed number of bytes.
- Dynamic storage format is used if a MyISAM table contains any variable-length columns (`VARCHAR`, `VARBINARY`, `BLOB`, or `TEXT`) or if the table was created with the `ROW_FORMAT=DYNAMIC` table option.
  - Dynamic format is a little more complex than static format because each row has a header that indicates how long it is. A row can become fragmented (stored in noncontiguous pieces) when it is made longer as a result of an update.
- For details about static versus dynamic table characteristics, see the MySQL documentation at:
  - http://dev.mysql.com/doc/refman/5.5/en/static-format.html
  - http://dev.mysql.com/doc/refman/5.5/en/dynamic-format.html

Quiz

Which of the following is *not* a reason to use fixed-width columns in the entire MyISAM table definition?

a. Disk space savings
b. Easier for indexes to reference the data file
c. Easier to fix if corruption occurs
d. Usually faster due to less row fragmentation

Answer: a

Optimizing MyISAM

- Run `OPTIMIZE TABLE` regularly.
- Disable indexes when deleting, inserting, or updating large amounts of data.
  - `ALTER TABLE t DISABLE KEYS` disables all nonunique keys.
  - `ALTER TABLE t ENABLE KEYS` reenables all nonunique keys and re-creates missing indexes.

Optimizing MyISAM

`OPTIMIZE TABLE`

- This command:
  - Removes fragmentation
  - Reclaims free space
  - Removes holes to allow concurrent insert to work

Disable indexes

- It may be more efficient to delete the indexes and then add them back after the large data task is completed.
- This is especially important in data warehouse build and load optimizations.

Practice 7-1 Overview: Optimize MyISAM

In this practice, you evaluate the effect of disabling indexes during the loading of large amounts of data into a MyISAM table, along with evaluating the effect of deleting multiple records in MyISAM tables. This practice covers the following topics:

- Creating and loading approximately 2 million records into a table in the `many_tables` database
- Deleting a large number of records in a MyISAM table and recording if the size of the table and indexes on disk are affected
- Optimizing the MyISAM table with deleted records and recording if the size of the table and indexes on disk are affected

ORACLE

Quiz

Which of the following is *not* a reason to run the `OPTIMIZE TABLE` command on a MyISAM table?

a. Removes holes to allow concurrent insert to work
b. Removes fragmentation
c. Reclaims free space
d. Improves secondary index

Answer: d

Minimizing MyISAM Table Lock Issues

- Remove “holes” in MyISAM data.
  - Allows concurrent insert
  - Execute `OPTIMIZE TABLE` on MyISAM tables regularly for applications that have a high number of deletes.
- Use `INSERT DELAYED` on noncritical data.
- Run multiple batches instead of one large delete operation.
- Optimize queries that cause blocks on the data.
- Set the `low_priority_updates` system variable to `TRUE`.

Minimizing MyISAM Table Lock Issues

`INSERT DELAYED`

- Statements waiting to execute using `INSERT DELAYED` are lost on an unexpected shutdown event.

Multiple small delete operations

- For performance purposes, do one large delete with a table-level lock.
- To prevent starving of other users, do multiple batches of smaller set of deletes or updates.
- Run multiple small delete operations versus one large operation:
  - `DELETE FROM t WHERE status="deleted" LIMIT 100;`

`low_priority_updates` system variable

- This system variable places `SELECT` statements at a higher priority than `UPDATE`, `DELETE`, `REPLACE`, and `INSERT` operations.
- Can be done with individual statements by adding `LOW_PRIORITY` after the statement command:
  - `UPDATE LOW_PRIORITY …`
- There is the danger that any non-`SELECT` statements can be starved and never execute.

Minimizing MyISAM Table Lock Issues

- Use vertical and horizontal partitioning.
  - Use vertical partitioning to separate the data you regularly update.
  - Horizontal partitioning helps with `ALTER TABLE` and `OPTIMIZE TABLE` commands.
- Consider using InnoDB tables.

Practice 7-2 Overview: MyISAM Table Locks

In this practice, you evaluate the effects of MyISAM table locking on delete, insert, and select operations. The practice covers simultaneously executing delete, insert, and select operations to evaluate the effect of lock contention on the operations.

ORACLE

MyISAM Settings

- `bulk_insert_buffer_size` sets the size of the MyISAM special tree-like cache used for bulk inserts.
  - Improves performance with bulk inserts of 100 to 1000+ records in a single insert statement
- `myisam_sort_buffer_size` sets the size of the buffer that is allocated when sorting MyISAM indexes.
  - This buffer is used when creating indexes with `CREATE INDEX` or `ALTER TABLE`, or with `REPAIR TABLE`.

MyISAM Settings

`bulk_insert_buffer_size` system variable

- Default is 8 MB.
  - Setting at `0` disables this buffer.
- Bulk inserts include the following commands:
  - `INSERT … SELECT …`
  - `INSERT … VALUES (…),(…),…`
  - `LOAD DATA INFILE`

`myiam_sort_buffer_size` system variable

- Default is 8 MB.
  - Limited to 4 GB on 32-bit systems (and 64-bit Windows systems)
- Set to the same size as the largest index size in your table.

MyISAM Settings

- `myisam_max_sort_file_size` sets the maximum size of the temporary file that MySQL is permitted to use while re-creating a MyISAM index.
  - Indexes are re-created during `ALTER TABLE`, `LOAD DATA INFILE`, and `REPAIR TABLE` operations.
- `myisam_recover_options` sets the MyISAM storage engine recovery mode.
  - Any combination of the values of `DEFAULT`, `OFF`, `BACKUP`, `FORCE`, and `QUICK`
    - You should separate multiple options with commas.

MyISAM Settings (continued)

`myisam_max_sort_file_size` system variable

- Default is 2 GB.
- If space is available, set this system variable to handle all the indexes in your table.
  - The key cache is used if there is not enough space, which would result in a much slower performance.

`myisam_recover_options` system variable

- For more information about the `myisam_recover_options` system variable, see the MySQL documentation: http://dev.mysql.com/doc/refman/5.5/en/server-options.html#option_mysqld_myisam-recover-options

Quiz

Which MyISAM server setting sets the maximum size of the temporary file that MySQL is permitted to use while re-creating a MyISAM index?

a. `bulk_insert_buffer_size`
b. `myisam_max_sort_file_size`
c. `myisam_recover_options`
d. `myisam_sort_buffer_size`

Answer: b

MyISAM Key Cache

- `key_buffer_size` sets the size of the buffer used for index blocks.
  - For MyISAM-only servers, set from 25% to 33% of memory.
- Use multiple key caches.
  - Consider using three key caches for busy servers:
    - A “hot” key cache: 20% of `key_buffer_size`
    - A “cold” key cache: 20% of `key_buffer_size`
    - A “warm” key cache: 60% of `key_buffer_size`
  - Creating key caches:
    - `SET GLOBAL hot_cache.key_buffer_size = 2G;`
    - `CACHE INDEX db1.tbl1, db1.tbl2 IN hot_cache;`

ORACLE

MyISAM Key Cache

`key_buffer_size` **system variable**

- For more information about the `key_buffer_size` system variable, see the MySQL documentation at http://dev.mysql.com/doc/refman/5.5/en/server-system-variables.html#sysvar_key_buffer_size.

Multiple key caches

- "Hot" key cache
  - Assign this key cache to the tables that are primarily used for searches (minimal updates).
- "Cold" key cache
  - Assign this key cache to the tables that are primarily used for updates, inserts, and deletes.
  - Your temporary tables are good candidates to use this key cache.
- "Warm" key cache
  - Assign this key cache to tables by default.
- For more information about using multiple MyISAM key caches, see the MySQL documentation at http://dev.mysql.com/doc/refman/5.5/en/multiple-key-caches.html.

MyISAM Key Cache

- Preload index into cache:
  - Performs a sequential read, which results in better performance
  - `LOAD CACHE INTO tbl1, tbl2 IGNORE LEAVES;`

MyISAM Key Cache (continued)

Preloading an index

- For more information about preloading an index, see the MySQL documentation at http://dev.mysql.com/doc/refman/5.5/en/load-index.html.

Midpoint insertion strategy

- For more information about the midpoint insertion strategy, see the MySQL documentation at http://dev.mysql.com/doc/refman/5.5/en/midpoint-insertion.html.

MyISAM Key Cache: Status Variables

- `Key_blocks_not_flushed` displays the number of key blocks in the key cache that have changed but have not yet been flushed to disk.
  - These blocks need to be flushed on shut down.
- `Key_blocks_used` displays the number of used blocks in the key cache.
  - Indicates the maximum number of blocks that have ever been in use at one time
- `Key_blocks_unused` displays the number of unused blocks in the key cache.
  - Can be used in conjunction with `key_cache_block_size` and `key_buffer_size` system variables to determine the amount of the key buffer that is being used

MyISAM Key Cache: Status Variables

`Key_blocks_not_flushed` status variable

- Blocks not flushed are lost in an unexpected shutdown event.

`Key_blocks_used` status variable

- Compare to `key_buffer_size` system variable.
  - Monitor this variable over a period of hours. If this status variable is considerably lower than `key_buffer_size`, decrease `key_buffer_size` accordingly.

`Key_blocks_unused` status variable

- The fraction of the key buffer being used can be determined by using the following formula:

```
1 - ((Key_blocks_unused * key_cache_block_size) /
   key_buffer_size)
```

MyISAM Key Cache: Status Variables

- `Key_read_requests` displays the number of requests to read a key block from the cache.
- `Key_reads` displays the number of physical reads of a key block from disk.
  - If `Key_reads` is large, increase `key_buffer_size`.
- `Key_write_requests` displays the number of requests to write a key block to the cache.
- `Key_writes` displays the number of physical writes of a key block to disk.
  - Ways to improve key cache write miss rate:
    - Bundle writes in SQL statements, `LOAD DATA` …, and so on.
    - Set `delay_key_writes` system variable to `ALL`.

MyISAM Key Cache: Status Variables (continued)

`Key_read_requests` and `Key_reads` status variables

- Calculate the cache miss rate by using these two variables:
  `Key_reads` / `Key_read_requests`
- Evaluate `Key_reads`/sec against the I/O system of your O/S.

`Key_write_requests` and `Key_writes` status variables

- Calculate the write miss rate by using these two variables:
  `Key_writes` / `Key_write_requests`
- Use the `delay_key_writes` system variable to identify how and if key writes are delayed.
  - `ALL` delays the key writes for all MyISAM tables.

Practice 7-3 Overview: Key Cache Effectiveness

In this practice, you evaluate key cache effectiveness by monitoring the `key_%` status variables. The practice covers the following topics:

- Modifying the `key_cache_buffer_size` server variable to a comparatively small value
- Executing multiple simple `SELECT` queries and calculating the effectiveness of the key cache by viewing the `key_blocks_unused` status variable

ORACLE

Quiz

Which MyISAM key cache status variable displays the number of physical writes of a key block to disk?

a. `key_blocks_used`
b. `key_reads`
c. `key_write_requests`
d. `key_writes`

Answer: d

MyISAM Full-Text Search

- Simple to use
  - … MATCH (*column/columns*) AGAINST ('*keyword*')
- Search with relevance or IN BOOLEAN MODE.
- Works only with MyISAM tables
  - Creating a MyISAM shadow table for InnoDB tables is possible.
- Indexing issues:
  - Indexes are updated immediately, causing slow updates.
  - Indexes that do not fit in memory cause slow performance.
  - Indexes are stored in BTREE.
- Performance suffers with high-frequency word searches.
  - Rewrite queries to improve performance.

MyISAM Full-Text Search

Use shadow tables when working with InnoDB tables and MyISAM tables.

- Bulk updates to a shadow search table can improve performance.

BTREE indexes

- Fetching data requires random I/O.
- Running OPTIMIZE TABLE on a regular basis improves performance.

Different searches and indexes

- Searching different columns requires different indexes.
  - A separate index is required for each distinct set of columns to be searched.
- If you set up a full-text index with multiple columns, you need to set up additional indexes to search the individual columns separately.

Common word searches

- Rewrite the following query to improve performance:

```
… MATCH (product) AGAINST ("video evita" IN BOOLEAN MODE)
```

Rewrite it as follows:

```
… MATCH (product) AGAINST ("evita" IN BOOLEAN MODE)
   AND product LIKE "%video%"
```

MyISAM Full-Text Search: Tips

- Cache only those full-text searches that are frequent.
  - Small-load full-text searches perform best.
- Skip `COUNT(*)` computations.
  - Very slow operations
  - Includes `SELECT SQL_CALC_FOUND_ROWS` … statements

MyISAM Full-Text Search: Tips

- `SQL_CALC_FOUND_ROWS`
  - To obtain this row count, include a `SQL_CALC_FOUND_ROWS` option in the `SELECT` statement, and then invoke `FOUND_ROWS()`:

```
mysql> SELECT SQL_CALC_FOUND_ROWS * FROM tbl_name
    -> WHERE id > 100 LIMIT 10;
mysql> SELECT FOUND_ROWS();
```

  - The second `SELECT` returns a number indicating how many rows the first `SELECT` would have returned had it been written without the `LIMIT` clause.

Practice 7-4 Overview: Full-Text Indexing

In this practice, you create a full-text index and compare the difference in using `NATURAL LANGUAGE` and `BOOLEAN` modes by:

- Creating a table that contains a full-text index
- Searching against the `fulltext` table using `NATURAL LANGUAGE` and `BOOLEAN` mode using multiple `SELECT` statements

ORACLE

Summary

In this lesson, you should have learned how to:

- Explain the best uses for the MyISAM storage engine
- Design tables to take advantage of using MyISAM
- Optimize MyISAM tables
- Describe MyISAM table locks
- Edit the MyISAM server settings
- Describe the MyISAM key cache
- Explain the MyISAM key cache status variables
- Use MyISAM full-text indexing capabilities

8

Other MySQL Storage Engines and Issues

Objectives

After completing this lesson, you should be able to:

- Improve the storage of large objects in MySQL
- Explain the best uses for the `MEMORY` storage engine
- Improve the performance of tables by using the `MEMORY` storage engine
- Describe the advantages of designing applications that use multiple storage engines
- Describe the advantages of designing applications that use a single storage engine

ORACLE

Large Objects

- Use InnoDB tables.
  - InnoDB skips `BLOB` columns if they are not requested in a `SELECT` statement.
- Separate out `BLOB` columns into separate tables.
  - MyISAM only
- Store large objects in files instead.
  - Memory consumption in MySQL is three times the size of the `BLOB`.
  - MySQL always performs full reads on `BLOB` columns.

Large Objects

Fragmentation: Watch out for fragmentation when deleting or updating large objects.

Practice 8-1 Overview: Large Objects

In this practice, you evaluate the effects of large objects on InnoDB and MyISAM storage engines by:

- Using the `PT_Stress.php` script to execute a query that displays a large-object column for every record in InnoDB and MyISAM storage engines and comparing the results
- Using the `PT_Stress.php` script to execute a query that displays a variable-character column for every record (that also includes a large-object column) in InnoDB and MyISAM storage engines and comparing the results

ORACLE

Quiz

Which storage engine skips `BLOB` columns if they are not requested in a `SELECT` statement?

a. CSV
b. InnoDB
c. `MEMORY`
d. MyISAM

Answer: b

MEMORY Storage Engine


MEMORY

- Content is lost on power failure.
- Hash and B-tree indexes only
- Table-level locking
- Fixed-length rows
  - Empty `VARCHAR(200)` columns still take up a lot of space.
- Lookups are very fast.
- The `max_heap_table_size` system variable limits the size of individual memory tables.

`MEMORY` Storage Engine: Uses

- Tables that are used for caching
- Temporary tables
- Tables that are used for buffering
  - Insert or update buffering
- Applications that require complex queries to be run
- Reporting on subsets of data
  - Daily work activity reports, quarterly reports, and so on

MEMORY Storage Engine: Performance

- Avoid table locks.
- Watch the size of rows.
  - Fixed size rows can require much more memory for your data.
- Do not exceed system memory.
  - MEMORY tables that need to be swapped to disk are slower than MyISAM tables.
- Use B-tree indexes for data with many duplicates.
  - Deletes on hash-indexed data are very slow.
  - Hash indexes also do not support ranges or prefix matches.

Practice 8-2 Overview: `MEMORY` Storage Engine

In this practice, you evaluate the `MEMORY` storage engine for performance and memory consumption by:

- Creating the `city_huge` database based on the MyISAM storage engine and running scripts against it to test performance and memory consumption
- Creating the `city_huge` database based on the `MEMORY` storage engine and running scripts against it to test performance and memory consumption

ORACLE

Quiz

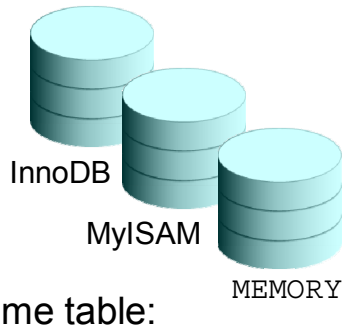
Which level of locking takes place on a table using the `MEMORY` storage engine?

a. Column
b. Page
c. Row
d. Table

Answer: d

Using Multiple Storage Engines: Advantages

- Each storage engine has different strengths and defined purposes.
  - Use MyISAM for constant data.
  - Use InnoDB for dynamic critical data.
  - Use `MEMORY` for temporary tables.
- Ability to change the storage engine:
  - Try out different storage engines on the same table:
    - `ALTER TABLE tbl ENGINE=<engine>`
  - Should only be done on nonprocession servers
  - Good tool for benchmarking data



Using Multiple Storage Engines: Advantages

Changing Storage Engine

Changing a table's storage engine requires the entire table to be copied. For large tables, this process can be time consuming. Determining the best storage engine during the development of the schema design saves you the trouble of changing the storage engine later.

Using a Single Storage Engine: Advantages

- Mixed database configuration can be more complicated to maintain.
  - Backups
  - Maintenance
  - Tuning
- The optimizer has a difficult time reporting correct plans when multiple storage engines are referenced.
- Mixing transactional tables with nontransactional tables in a transaction can cause replication difficulties.

Using a Single Storage Engine: Advantages

Note: For details about replication and possible difficulties due to mixed storage engines, see http://dev.mysql.com/doc/refman/5.5/en/replication-features-transactions.html.

Quiz

Mixing transactional tables with nontransactional tables in a transaction can cause replication difficulties.

a. True
b. False

Answer: a

Summary

In this lesson, you should have learned how to:

- Improve the storage of large objects in MySQL
- Explain the best uses for the `MEMORY` storage engine
- Improve the performance of tables by using the `MEMORY` storage engine
- Describe the advantages of designing applications that use multiple storage engines
- Describe the advantages of designing applications that use a single storage engine

9

Schema Design and Performance

Objectives

After completing this lesson, you should be able to:

- Describe the purpose of designing a schema based on the data to be stored
- Describe how normalization can affect performance
- Use the appropriate data type to improve performance
- Use indexing to improve performance
- Describe the different indexing types
- Use partitioning to improve query performance

Schema Design Considerations

- Reports or output that the database will be responsible for
  - Identify queries that need to be executed.
- Type of data that needs to be collected to create the reports or output
  - Determine data types and sizes needed.
  - Identify relationships that exist.
- Best way to store the data
  - Identify data that needs to be housed in the database.

Schema Design Considerations

Reports and output

- Understanding what end users need is an important step in designing the correct schema design.
  - You can determine this by understanding the problem that the application is designed to solve.

Data collected

- You can understand the type of data needed to meet end-user requirements by defining:
  - The data type and size for the data to be collected
  - The relationships and cardinality that exist within the data pool

Storing data

- With an understanding of the data, you can determine if the database is the best place to store all of the data.
  - Images or other binary files can produce better performance if stored outside of the database.
  - XML or other data formats might be better handled by the applications that retrieve data from the database instead of being stored with the data.

Schema Design Tasks

- Normalize the data.
  - Eliminates redundant data
  - Provides flexible access to data
  - Minimizes data becoming inconsistent
- Choose the correct data types and size.
  - Improves performance
  - Protects data
- Create the most efficient indexes.
  - Improves data processing

Schema Design Tasks

Normalizing the data

- There are many levels to normalization that you can work through when determining the best way to store your data. However, over-normalizing is just as bad as under-normalizing data. It is important to remember that the ultimate purpose of normalization is to get accurate data to the end user as quickly as possible. If you are normalizing just for the sake of normalizing, you are most likely going to over-normalize your database.

Data types and size

- Choosing the correct data type can be considered a minor task in the schema design process. However, one of the biggest factors in database performance is the actual storing of data. Small amounts of extra data that are stored in an individual column because of an improper data type or size can become a large problem when the data grows. The result is an exponential growth in the file system needs and, more importantly, the memory that is used.

Efficient indexes

- Indexing, like normalization, can produce large gains in performance if the number and types of indexes are created properly. Not using an index on a table or under-indexing a table can result in full table scans, while over-indexing a table can result in a query that has to interact with more indexes than are actually needed to produce the results. Both of these extremes must be resolved to see an improvement in performance by indexing tables.

Quiz

Which of the following is *not* a schema design task?

a. Choosing the correct data types and sizes
b. Choosing the best monitor to view the data
c. Creating the most efficient indexes
d. Normalizing the data

Answer: b

Normalization and Performance

- All “objects” are in their own tables. There is no redundancy.
- Normalization provides a compact, single update location to modify objects
- Normalization results in performance degradation when joining tables
- The optimizer is limited in the choices it can make for the selection and sorting data.
- Online transaction processing (OLTP) applications can gain the most from having normalized data.

Non-Normalization and Performance

- The size of tables containing non-normalized data can be large.
- Updating data can have added complexity.
- Data loss is more likely.
- With a limited need for joins, the optimizer has more choices.
- Decision support system (DSS) applications can gain the most from non-normalized data.

Mixing Normalization and Performance

- Benefits can be gained from caching some static data in the table.
- Having some non-normalized data can result in performance gains.
- Include non-normalized value as key for simple objects.
- Non-normalized tables can simulate materialized views.
- Reference non-normalized data by `PRIMARY` or `UNIQUE` key.
  - MySQL can optimize these by prereading values that are constant.

Mixing Normalization and Performance

Non-normalized data and better performance

- You must consider the cost of updates to ensure that you are not creating a performance problem.

Non-normalized data as key for simple objects

- Example: Adding an IP address to a table as a key (even though it may exist in another table) can limit the need to join tables and can produce substantial performance gains.

Materialized views

- MySQL does not support materialized views.
  - Materialized views are different from normal views. They are not "updated" when the original tables of the `SELECT` query are modified.
- Creating a table that contains non-normalized data on an interval basis can simulate a materialized view in MySQL.
  - Create a normal view based on the `SELECT` query that you want to store.
  - Create a materialized view table with the indexes that support your needs (based on the normal view created).

- Run the following commands on a regular interval (at midnight, every six hours, or at other regular times):

```
LOCK TABLES mView WRITE;
TRUNCATE mView;
INSERT INTO mView SELECT * FROM nView;
UNLOCK TABLES;
```

Practice 9-1 Overview: Schema Design

In this practice, you evaluate the effects of normalized and non-normalized data in a variety of scenarios by:

- Using the `PT_Stress.php` script to execute queries that draw data from normalized and non-normalized tables that do not require the joining of normalized tables
- Using the `PT_Stress.php` script to execute queries that draw data from normalized and non-normalized tables that require the joining of normalized tables

ORACLE

Quiz

What are some of the advantages to mixing normalization in a database application? (Choose all that apply.)

a. Redundancy is completely eliminated.
b. Non-normalized data used as a key for simple objects can prevent the need to join tables.
c. Non-normalized tables can simulate materialized views.
d. The process to update data is simplified.

Answer: b, c

Data Types

- Use the correct data type for the data being stored.
  - Incorrect data types can cause problems beyond performance.
- Set the appropriate length for a data type
  - Larger data types than are necessary lead to poor performance.
- Use `NOT NULL` in columns that do not require a `NULL` value.
- Use an appropriate character length.
  - Some buffers are fixed in memory.
  - Sorting files and temporary tables are fixed length.
- Converted data can result in incorrect data types.

Data Types

Correct data types

- True numeric values should be stored as numbers.
  - `09` and `9` represent the same number but are different strings.
  - Such things as postal codes and phone numbers are not true numeric values and are best stored as strings.
- Date values should be stored as dates.
  - It is best to store "raw" data in the database. Storing formatted or manipulated data in a table can lead to eventual problems.

Appropriate data type length

- Examples:
  - Most date-of-birth (DOB) columns do not require the end user to know the exact time the individual was born. Only the month, day, and year are necessary.
  - Using `DATE` rather than `DATETIME` can reduce the storage requirements by 5 bytes for each record. In a table that houses 100,000 records, that can save 500,000 bytes from having to be stored and retrieved.

Converted data

- Having control over data that is loaded is essential to ensuring that loaded data does not lose any of its value on conversion.

Practice 9-2 Overview: Data Types

In this practice, you evaluate the effects of data types in a variety of scenarios by:

- Viewing the size of specific tables in the `/var/lib/mysql/isfdb` directory
- Modifying a `DATE` column to a `DATETIME` column and determining the effect on the size of the table
- Creating multiple copies of the same table that contain different string data types and determining the effect on the size of the table
- Using the `PT_Stress.php` script to execute queries against the different copies of the table to determine the effect that string data types have on performance

ORACLE

Quiz

What are some best practices when determining the data type for a column? (Choose all that apply.)

a. Use the correct data type for the data being stored.
b. Use the largest length possible for a data type.
c. Use `NOT NULL` in columns that do not require a `NULL` value.
d. Worry about converted data after it has been imported into your tables.

Answer: a, c

Indexes

- Indexes can speed up the retrieval of data.
  - Indexes can be extensive to maintain.
- Only the prefix of a composite index can be used alone.
  - The order of columns in a composite index is important.
- Minimize the number of indexes created on a column.
  - It is best to have only one index for a column.
- Indexes that are unique should be defined as `UNIQUE`.
- Apply indexes on columns that contain nonrepeating or selective data.
- Remove indexes that are dead.
  - Indexes that are not being used continue to drain performance during DML operations.

Indexes

Improve speed

- Indexes speed up reads and lookups either by index-only reads for `SELECT` statements or by the `WHERE` clause for `SELECT/UPDATE/DELETE` statements.
  - Indexes increase the time for write operations (`INSERT/UPDATE/DELETE` statements) because every index must be updated.
- Indexes can speed up the sorting of some `ORDER BY` queries.

MySQL composite indexes

- When an index is created from two or more columns, only the first column of the index can be used alone.
  - `KEY (a,b)`
    - `a=4` would use index
    - `a=4 AND b=5` would use index
    - `b=5` would not use index
- The order of columns, especially in a `BTREE` index, is important.

Two or more indexes

- Having two or more indexes on the same column can result in unnecessary performance overhead.
- This includes having a stand-alone index on a column `( KEY(a) )` and that same column being part of a prefix index `( KEY(a,b))`.

Indexes

Unique indexes

- Unique indexes should be created on columns that contain no duplicate values.
- An index that is defined as `UNIQUE` provides the query optimizer with more opportunities to create an effective execution plan.

Index Selectivity

- The selectivity of an index is defined as the percentage of rows in a table that have the same value for the indexed column.
  - `SELECT COUNT(DISTINCT(author_lastname)) FROM authors` results in 28,920.
  - `SELECT COUNT(author_lastname) FROM authors` results in 68,700.
  - The selectivity of the index for this column is calculated as 28,920/68,700, which is 42%.
    - An index selectivity of 33% or better is preferred.

Indexing

- Covering indexes can be accessed without reading the original table.
- Prefix an index if the data is selective enough by the first few characters.
- Using short keys improves the performance of the index.
  - Columns using the integer data type are the best to index.
- Key values that are close to each other produce the best performance.
  - An indexed `AUTO_INCREMENT` field results in better performance than an indexed `UUID()` field.

Indexing

Covering indexes

- If a `SELECT` query references only those fields that are contained entirely within the index, MySQL may skip accessing the original table entirely.
  - ... `UNIQUE KEY 'rental_date' ('rental_date', 'inventory_id', 'customer_id')` ...
  - `SELECT inventory_id, customer_id FROM rental WHERE rental_date > '2005-05-25'` results in the `rental_date` index being used without the need to locate (seek) and read the full row.

Prefix indexes

- Columns that contain lengthy text can be indexed based on a number of beginning characters in the column.
  - This reduces the size of the index on disk, as well as when it is subsequently stored in memory.
  - The `titles.title_title` column is assigned as a `MEDIUMTEXT` data type.
  - The index using the `titles.title_title` column stores the first 50 characters. With the titles table containing over 500,000 records and over 25,000 title_title columns containing text lengths greater than 50 characters (with a maximum character length of 931), limiting the length of the index can save a large amount of disk space.

Indexing

- Executing `OPTIMIZE TABLE` compacts and sorts indexes.
- Executing `ANALYZE TABLE` updates the key distribution statistics.
  - These key distribution statistics are used by MySQL to assist in:
    - Ordering joined tables
    - Choosing the most efficient index for a query

Index Types

- BTREE is supported by all MySQL storage engines.
  - Default except for MEMORY (which defaults to HASH)
  - Assists equal and range queries along with sorting operations
- HASH is supported by the MEMORY storage engine only.
  - Provides a faster, smaller footprint than any other index types
- RTREE is supported by the MyISAM storage engine only.
  - Works with GIS data
- FULLTEXT is supported by the MyISAM storage engine only.
  - Speeds up natural language searches

Index Types

BTREE

- A BTREE index on KEY (a,b) would help with the following SELECT statements:
  - SELECT id FROM table WHERE a = ? AND b = ?
  - SELECT id FROM table WHERE a = ? AND b BETWEEN ? AND ?
  - SELECT id FROM table WHERE a = ? AND b > ?
  - SELECT id FROM table WHERE a = ? ORDER BY b
  - SELECT id FROM table WHERE a = ? AND b LIKE 'prefix%'
  - SELECT id FROM table WHERE a = ?
  - SELECT id FROM table ORDER BY a

HASH

- Slower; many nonunique values

RTREE

- Speeds up multidimensional lookups (such as find POINT in region)

FULLTEXT

- Slows down update for long texts

Practice 9-3 Overview: Indexes

In this practice, you observer the effects of indexes in a variety of scenarios by:

- Viewing the size of specific tables in the `/var/lib/mysql/isfdb` directory
- Creating a `MEMORY` table with both `BTREE` and `HASH` indexes
- Using the `PT_Stress.php` script to run multiple queries against the `MEMORY` table created to test the performance using the different index types
- Creating a table that uses a prefix index and evaluating the performance along with the size difference between a full index and a prefix index

ORACLE

Quiz

Executing which of the following commands will compact and sort indexes on the city table?

a. `EXPLAIN TABLE city;`

b. `ORDER INDEX city;`

c. `SHOW TABLES LIKE 'city';`

d. `OPTIMIZE TABLE city;`

Answer: d

Partitioning

- Partitioning can improve query efficiency.
  - Partitioning pruning is possible.
- Proper partitioning can result in fewer table management and maintenance tasks.
- Partitions can be distributed across different file systems and hardware.
  - Different storage engines perform better than others in this setup.

Partitioning

Partition pruning: Refers to the MySQL Optimizer removing entire partitions from a query execution plan, which results in less data that is read.

Partitioning and the File System

- Partitioning operations are dependent on the file system.
- MySQL system variables to consider:
  - `open_files_limit`: Ensure that the size of this limit is set properly to prevent MySQL from running out of file descriptors.
  - `myisam_max_sort_file_size`: Ensure that this is set high enough to use "Repair by sorting" versus "Repair by Key Cache" when running `ALTER` statements.
  - `innodb_file_per_table`: Turning this variable on can improve partitioning operations.

Partitioning and the File System

Partitioning performance depends on such factors as:

- Type of file system being used
- Characteristics of the file system
- Speed of the disk
- Amount of swap space available
- Efficiency of the file system

System variables

- `open_files_limit`
  - The MySQL server implementation of partitioning can be described as multiple tables housing data that correlates from table to table.
  - The `open_files_limit` system variable, if set too low, results in MySQL errors when your database has multiple tables.
  - Because partitioning increases the number of “tables” that MySQL sees in your database, this setting needs to be set high enough to account for MySQL's implementation of partitioning.
- `innodb_file_per_table`
  - When this system variable is turned on, each partition has its own `InnoDB “table” .ibd` file that stores data and indexes.

Other Partitioning Performance Issues

- Using a `WHERE` clause that allows partition pruning is usually faster than a nonpartitioned table.
- Altering and maintaining partitions is usually faster than altering the whole table.
- Creating useful indexes is just as important with partitioned tables.
  - You need to consider partition pruning when designing indexes on partitioned tables.
- Per-partition key caches are supported.
  - MyISAM tables support key caching on one partition, several partitions, or all partitions.

ORACLE

Other Partitioning Performance Issues

Altering and maintaining partitions

- The only case when this is not faster is when the server needs to alter all partitions.
  - Common for `HASH/KEY` partition management, which can be avoided by using `LINEAR HASH/LIST`.
- Dropping or truncating a partition is as fast as removing one file, while deleting the same amount of data would take a long time and have different results depending on the storage engine.

Partitioning Limitations

- Foreign keys are not supported.
- `ALTER TABLE … ORDER BY` operations result in only the rows of individual partitioned "subtables" being ordered.
- `FULLTEXT` indexes are not supported.
- Spatial columns are not supported.
- Temporary tables cannot be partitioned.
- There are limitations on the data type of the column that can be used for partitioning.
- Subqueries cannot be used as a partitioning key.
- Not all functions are allowed as partitioning functions.

Partitioning Limitations

Data type limitations

- `INTEGER` columns are required for most partitioning types.
  - An expression that results in an integer is also acceptable.
  - Although `NULL` values are accepted, you should avoid them in columns used in partitioning.
    - See http://dev.mysql.com/doc/refman/5.5/en/partitioning-handling-nulls.html if you must have `NULL` values in a column that is used for partitioning.
- For `KEY` partitioning: MySQL's internal key-hashing functions produce a data type that is allowed in partitioning.
- For `RANGE` and `LIST` partitioning: The string, `DATE`, and `DATETIME` columns are acceptable.
  - `BLOB` or `TEXT` data types cannot be used as columns for partitioning.

Functions

- Only the `YEAR()`, `TO_DATE()`, `TO_SECOND()`, and `UNIX_TIMESTAMP()` functions are optimized for range partitioning.

Practice 9-4 Overview: Partitioning

In this practice, you evaluate how partitioning can improve performance. The practice covers the following topics:

- Creating a copy of an existing table that uses range partitioning
- Using the `PT_Stress.php` script to execute queries against the original table and partitioned table to compare performance

ORACLE

Quiz

Which of the following is *not* a limitation of MySQL partitioning?

a. Foreign keys are not supported.
b. Partitioning pruning is not supported.
c. Spatial columns are not supported.
d. Temporary tables cannot be partitioned.

Answer: b

Summary

In this lesson, you should have learned how to:

- Describe the purpose of designing a schema based on the data to be stored
- Describe how normalization can affect performance
- Use the appropriate data type to improve performance
- Use indexing to improve performance
- Describe the different indexing types
- Use partitioning to improve query performance

10

MySQL Query Performance

ORACLE

Objectives

After completing this lesson, you should be able to:

- Describe best practices for SQL writing and execution
- Read `EXPLAIN` output
- Adjust MySQL Optimizer behavior
- Locate problematic queries

SQL Tuning: General Best Practices

- Test on a realistic data set.
- Watch for changes to your data and requirements.
- Do not make ported code assumptions.
- Use efficient data types.
- Consider multiple queries.

SQL Tuning: General Best Practices

- Use a complete set of real-world data, not just a subset or sample of your data.
- As your database and applications grow, be aware that performance characteristics of the initial application can change.
- Test all ported code thoroughly. Do not assume a query that executes quickly on other databases can do so on your MySQL server.
- Keep data types at the smallest possible size that meet your data requirements.
- Break slow and complex queries into multiple, smaller queries.

Quiz

You should break slow, complex queries into multiple, smaller queries.

a. True
b. False

Answer: a

EXPLAIN

- The `EXPLAIN` command describes how MySQL will execute your particular SQL statement, but it does not return any data from the data sets.
- You can use the `EXPLAIN` command by preceding your `SELECT` statement with the keyword.
  - `EXPLAIN SELECT ...`
  - The `EXPLAIN` command produces a number of columns that provide information about how MySQL will execute the statement.

Quiz

Which script results in the MySQL server describing how it intends to execute your particular SQL statement?

a. `DEFINE SELECT * FROM world;`

b. `EXPLAIN SELECT * FROM world;`

c. `HOW SELECT * FROM world;`

d. `SELECT EXPLAIN * FROM world;`

Answer: b

EXPLAIN Output

- id: A simple identifier for your SELECT statement
- select_type: Describes the type of SELECT being performed
- table: Displays the names of tables that will be used in the access strategy
- type: Describes the strategy deployed by MySQL to access the data in the table or index
- possible_keys: Provides the available indexes that MySQL had to choose from in evaluating the access strategy
  - NULL is the output if there are no keys available.

EXPLAIN Output

type access strategy:

- Each access strategy is assigned a sliding performance scale that is compared to a number of statistics.
- The two most important statistics used are:
  - **Selectivity of an index:** This number identifies the cardinality (the number of unique values in the field) in the index tree, thus giving MySQL a possible number of keys that an index will match in the WHERE or ON conditions of the query.
  - **Sequential reads versus random seeks:** The relative speed of doing sequential reads for data on disk versus reading an index key and accessing table data using random seeks from the index row pointers to the actual data location

EXPLAIN Output

- `key`: Displays the actual key chosen to perform the data access
  - `NULL` is the output if there are no keys available.
- `key_len`: Provides the length (in bytes) of the chosen key
  - The length is `NULL` if the key column says `NULL`.
- `ref`: Displays the columns in the key that is used to access data in the table
- `rows`: Displays the number of rows that MySQL expects to find
- `Extra`: Extra information pertaining to this particular row's access strategy

ORACLE

EXPLAIN Output (continued)

- **`ref` value:** Produces a single constant value if the join has been optimized away
- **`rows` value:** Based on statistics that the MySQL server keeps on the table or index (key) chosen to be used, and also based on any preliminary calculations that the server has done based on the `WHERE` clause

Quiz

Which field in the `EXPLAIN` output describes the strategy deployed by MySQL to access the data in the table or index?

a. `keys`
b. `possible_keys`
c. `select_type`
d. `type`

Answer: d

EXPLAIN select_type

- SIMPLE: Normal, non-UNION, non-subquery SELECT
- PRIMARY: Outermost SELECT in any statement
- UNION: Second or later SELECT in a UNION
- DEPENDENT UNION: Second or later SELECT in a UNION
  - Dependent on the results of an outer SELECT statement
- UNION RESULT: Result of a UNION
- SUBQUERY: Primary SELECT in a subquery
- DEPENDENT SUBQUERY: Contains a reference to a table that also appears in the outer query
  - Also known as a *correlated* subquery
- DERIVED: FROM clause contains a subquery.

Quiz

Which `EXPLAIN select_type` value is also known as a *correlated* subquery?

a. `DEPENDENT UNION`
b. `DEPENDENT SUBQUERY`
c. `DERIVED`
d. `SUBQUERY`

Answer: b

EXPLAIN type

Order from most efficient access to least efficient:

- `system`: Used when your `SELECT` statement is requesting data from an in-memory table and the result has only one row of data
- `const`: Contains at most one matching row, which is read at the start of the query
- `eq_ref`: Contains a single row that matches rows from a previous table data set
- `ref`: Contains one or more rows that match rows returned from a previous table data set
- `ref_or_null`: Identical to `ref` with following exceptions:
  - `NULL` values: Key used contains `NULL` values.
  - `OR` null condition: Query contains a `WHERE` with an `OR` *key_column* that is `NULL`.

ORACLE

EXPLAIN type

- The `system` type access strategy is a special type of access strategy that MySQL can deploy based on the `const` join type.
- The `const` type access strategy meets one of the following criteria:
  - **Limited number of rows:** The table has no matching rows to provide for the query or one row of data that meets the query requirements.
  - **Unique key(s) are present:** The table contains a unique, non-nullable key for which a `WHERE` condition contains a single value that is present for each column in the key.
- The `eq_ref` access strategy type is used only if both of the following conditions are met:
  - **Keys used:** All parts of a key are used by the join in the query.
  - **Unique key(s) are present:** The tables contain a unique, non-nullable key for which a `WHERE` condition contains a single value that is present for each column in the key.

- The `ref` access strategy is used when either of the following occurs:
  - **Leftmost part of join is used:** The join condition uses only the left-most part of a multicolumn key.
  - **Non-unique and non-null key:** The key used is not unique but does not contain any null values.

EXPLAIN type

- `index_merge`: Multiple executions of `ref`, `ref_or_null`, or `range` accesses are used to retrieve key values matching various `WHERE` conditions.
- `unique_subquery`: A subquery is used in an `IN` clause in the `WHERE` condition.
- `index_subquery`: Identical to `unique_subquery` except the values returned from the subquery are not unique
- `range`: Contains a `WHERE` clause that uses range operators
- `Index`: Sequential scan is performed on all the key entries of an index.
- `ALL`: Sequential scan is performed on all data in the table.

ORACLE

EXPLAIN type (continued)

- The `index_merge` type access strategy is used when the results of the various retrievals are combined to form a single data set.
- The `unique_subquery` type access strategy:
  - Values returned from the subquery are unique.
  - *Child query* is another name for a subquery of this type.
- The `range` access strategy type:
  - Range operators include `>`, `>=`, `<`, `<=`, `ON`, `LIKE` and `BETWEEN`.
- The `index` access strategy is used when both of the following conditions are met:
  - **Slow data retrieval:** No `WHERE` clause is specified, or the table used contains no index that would speed up data retrieval.
  - **Covering index:** All columns in the `SELECT` statement for this table are available in the index.
- The `ALL` access strategy is used when either of the following conditions is met:
  - **No conditions on keys:** No `WHERE` or `ON` condition is specified for any columns of the table's keys.
  - **Poor index key distribution:** Poor index key distribution results in a sequential scan that is more efficient than numerous index lookups.

Quiz

Which `EXPLAIN type` value is considered the most efficient?

a. `ALL`
b. `const`
c. `index_merge`
d. `system`

Answer: d

EXPLAIN key

- Include index hints in the `SELECT` statement issued.
  - `USE INDEX (key_list)`: Use only one of the possible indexes from *key_list* to find rows in the table.
  - `IGNORE INDEX (key_list)`: Notifies MySQL not to use a particular index from *key_list*
  - `FORCE INDEX (key_list)`: Acts like `USE INDEX (key_list)` but with the addition that a table scan is assumed to be very expensive
- Include join hints in the `SELECT` statement issued.
  - `STRAIGHT_JOIN`: Forces MySQL to use the join order defined in the statement

EXPLAIN key

STRAIGHT_JOIN

- By default, MySQL adjusts the join order to improve query effectiveness.
- Adding this option overrides adjusting the join order by MySQL.

Quiz

Which `EXPLAIN key` value forces MySQL to use the join order defined in the statement?

a. `FORCE_INDEX`
b. `FORCE_JOIN`
c. `STRAIGHT_JOIN`
d. `USE_INDEX`

Answer: c

EXPLAIN Extra

- `Using where`: A `WHERE` clause is being used to restrict which rows to match against the next table or send to the client.
- `Using filesort`: A manual sorting-pass is needed.
- `Using temporary`: A temporary table is being used to store the result set at some stage of the execution.
- `Using index`: All operations on this table reference are completed in the index tree. There are thus no row lookups.

EXPLAIN Extra

Using filesort

- There is no extra information given to tell if the output will fit in memory or if it will use file-merge sort.

Using temporary

- There is no way to determine from the `EXPLAIN` output if the temporary table will use the `MEMORY` storage engine or if it will use the MyISAM storage engine.

EXPLAIN Extra

- `Using index for group-by`: An index is available to retrieve all columns of a `GROUP BY` or `DISTINCT` query.
  - No extra disk access to the actual table is needed.
- `Using sort_union(...)`, `Using union(...)`, and `Using intersect(...)`
  - These indicate how index scans are merged for the `index_merge` join type.
- `Distinct`: Searching stops after the first matching row is found.
- `Not exists`: A `LEFT JOIN` optimization has taken place.
- `Range checked for each record (index map: N)`

ORACLE

EXPLAIN Extra (continued)

`Not exists` output:

- MySQL was able to do a `LEFT JOIN` optimization on the query.
  - No more rows are examined in this table for the previous row combination after MySQL finds one row that matches the `LEFT JOIN` criteria.

`Range checked for each record (index map:N)` output:

- There are no good indexes to use from primary data.
- Some of the indexes may be useful after column values from preceding tables are known.

Quiz

Which `EXPLAIN Extra` value indicates that a `LEFT JOIN` optimization has taken place?

a. `Distinct`
b. `Not exists`
c. `Using index`
d. `Using_sort_union`

Answer: b

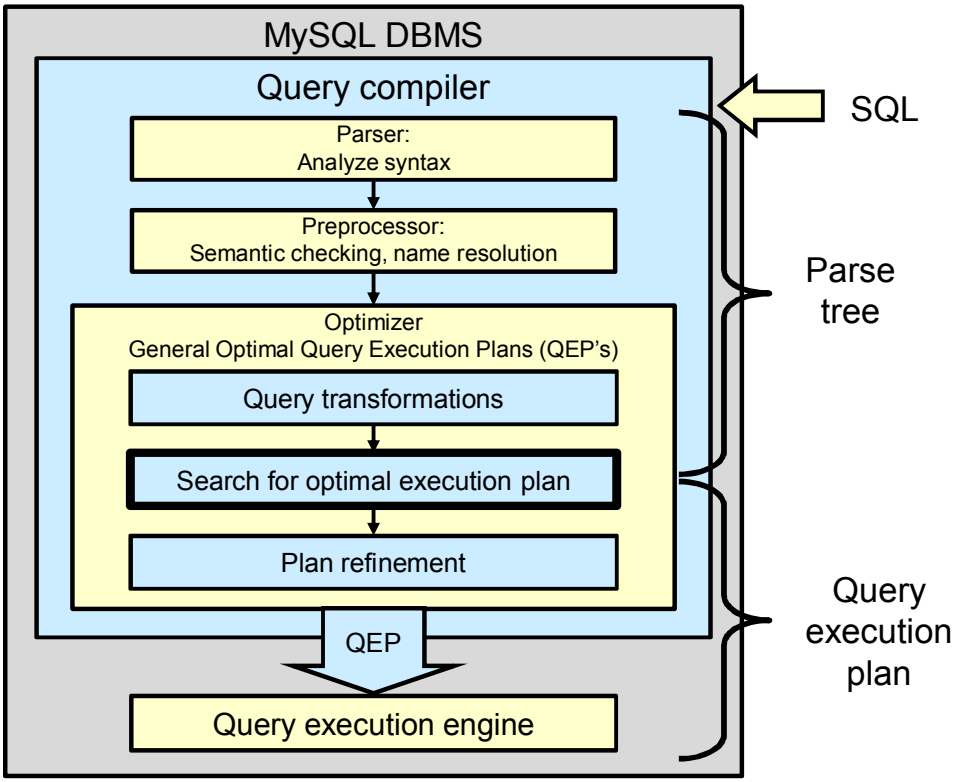
Practice 10-1 Overview: `EXPLAIN` Outputs

In this practice, you execute a number of `SELECT` statements by using the `EXPLAIN` command to review the different outputs that are possible. The practice covers the following topics:

- Executing multiple `SELECT` statements against the `world` database
- Adding indexes to existing tables
  - **Note:** Throughout this practice, there are numerous steps to clean up the `world` database and its tables to ensure that the database itself remains unaffected at the end of the exercises.

ORACLE

MySQL Query Optimizer



ORACLE

MySQL Query Optimizer

The task of the query optimizer in MySQL is to find the best or optimal plan for executing the SQL query that is submitted to the server. The query optimizer searches for the best plan among all the possible query evaluation plans.

A process called *greedy optimization* enables the optimization engine to significantly reduce the amount of time it spends calculating optimal execution paths by a process of intelligent best-path reductions. For larger, more complex queries, this had previously become a bottleneck in the server's performance.

Quiz

A process called *generous optimization* enables the optimization engine to significantly reduce the amount of time it spends calculating optimal execution paths by a process of intelligent best-path reductions.

a. True
b. False

Answer: b

Finding Problematic Queries

- Run `EXPLAIN`.
  - Monitor SQL statements being sent to the server.
  - Run these statements by using the `EXPLAIN` command.
- Enable the slow query log.
  - Query logs can be expensive on performance.
  - The slow query log is worth the minimal overhead.
- Enable the general query log.
  - Enable this periodically to collect a sampling of queries that are being sent to the server.

Finding Problematic Queries

Slow Query Log

- Queries that do not use indexes are logged in the slow query log if the `log-queries-not-using-indexes` system variable is turned on.
- The `--log-slow-admin-statements` server option enables logging of slow administrative statements such as `OPTIMIZE TABLE`, `ANALYZE TABLE`, and `ALTER TABLE` to the slow query log.
- Queries handled by the query cache are not added to the slow query log, nor are queries that would not benefit from the presence of an index because the table has zero rows or one row.

General Query Log

- The server writes information to this log when clients connect or disconnect, and it logs each SQL statement received from clients.
- This log can be very useful when an error in a client is suspected and it is beneficial to know exactly what the client sent to the server.

Finding Problematic Queries

- Run `SHOW PROCESSLIST`.
  - Provides a real-time list of all connections to the server.
  - Displays the action being performed by each connection.
  - Terminate processes that are problematic:
    - `KILL <connection_id>`
      - Terminates the connection itself
    - `KILL QUERY <connection_id>`
      - Terminates the query being executed by the connection

Quiz

Which of the following steps can assist you in finding problematic queries? (Choose all that apply.)

a. Run `EXPLAIN`.
b. Enable the slow query log.
c. Enable the general query log.
d. Run `SHOW PROCESSLIST`.

Answer: a, b, c, d

Practice 10-2 Overview: Improve Query Executions

In this practice, you improve the execution of a sample query by:

- Executing the sample `SELECT` statement against the `world` database
- Using the `EXPLAIN` command to evaluate the steps needed to reduce the negative performance issues associated with the query
- Implementing changes to the table to improve the performance of the query based on the outcome of your evaluation

ORACLE

Practice 10-3 Overview: Locate and Correct Problematic Queries

In this practice, you locate and correct queries that are executing longer than determined to be acceptable. The practice covers the following topics:

- Reviewing the current system variables associated with the slow query log
- Setting the system variables to control the types of queries to be captured by the slow query log
- Using the `mysqlslap` application to execute a predefined script that contains multiple SQL statements
- Reviewing the statements captured by the slow query log and performing the steps necessary to improve the statement execution times

ORACLE

Summary

In this lesson, you should have learned how to:

- Describe best practices for SQL writing and execution
- Read `EXPLAIN` output
- Adjust MySQL Optimizer behavior
- Locate problematic queries

11

Performance Tuning Extras

Objectives

After completing this lesson, you should be able to:

- Describe how your hardware and O/S can affect performance
- Demonstrate how to set up and evaluate logging to improve performance
- Describe best practices for backup and recovery

Hardware Considerations

CPU

- The standard is 64 bit.
- For large workloads, larger CPU cache improves performance.
- Multiple queries scale well for multiple CPUs.
  - Single query uses single CPU.
- Intel Hyper-Threading Technology has the potential to improve performance by as much as 20%.
- Dual-core CPUs provide up to 80% increase in CPU capacity.
- High-performance system bus minimizes overload on high load.

ORACLE

Hardware Considerations

64-bit standard

- The benefits of 64-bit systems include improved RAM addressing, faster CPU response time, improved parallel processing, and faster file I/O response time.

CPU Cache

- A large CPU cache provides improved performance.
  - 1 MB to 2 MB has the potential to improve performance by as much as 30%.
  - A large number of threads accessing the MySQL server can benefit from the larger CPU cache.

Intel Hyper-Threading Technology

- More information can be found at http://www.intel.com/technology/platform-technology/hyper-threading/index.htm.

Dual dual core versus dual quad core

- In most applications, dual dual cores at a faster clock speed usually perform better than dual quad cores at a slower clock speed.
- If your applications are highly concurrent and CPU-bound, dual quad cores have the potential to perform better.

System bus

- Having a high-performance system bus or several system buses spreading high bus usage devices (network, disk, and so on) has the potential to improve performance.

Quiz

The benefits of a 64-bit system include which of the following? (Choose all that apply.)

a. Improved RAM addressing
b. Faster CPU response time
c. Improved parallel processing
d. Faster file I/O response time

Answer: a, b, c, d

Hardware Considerations

- Memory
  - Bandwidth is a frequent bottleneck for CPU-bound workloads.
  - Fast memory, dual-channel memory, and a dedicated bus have the potential to provide multiple benefits.
- Network card
  - Minimize latency by choosing a Gigabit Ethernet card.
  - Choose a network interface card that supports CPU offloading.
  - Verify driver support for your O/S.
- Disks
  - Use RAID to ensure data security.
  - Use multiple channels when working with multiple devices.

ORACLE

Hardware Considerations (continued)

Memory

- Fast memory has the potential to improve cached disk performance.
- The increase in dual-channel memory performance has a minimal effect on overall system performance.
- A dedicated bus on a shared-memory multiprocessor (SMP) has the potential to improve performance.

Disks

- RAID10 is the best choice for many devices. RAID1 is a good choice if you have only two disks.
  - The software associated with these devices use very little CPU time for random I/O.
- RAID5 is slower for applications with random writes in addition to slower rebuild time.
- RAID0 can be used in slaves to improve performance.
- Use larger RAID chunk (256K to 1MB) to improve performance.
  - Chunk is the “atomic” mass of data that is written to the devices.
- Use write cache that is backed up by battery.
  - Ensures truly ACID transactions with minimal performance overhead.

Hardware Considerations

- Storage sharing
  - Storage area network (SAN) is easy to manage but slower than direct disks.
  - Network attached storage (NAS) works well with logs, binary logs, and read-only MyISAM.
    - Problems have been reported with InnoDB.
  - InnoDB logs can be placed on a dedication RAID1 drive if there are many devices.
    - O/S can use the same drive.
- Extending your hardware
  - Hardware should be purchased with expansion in mind.
    - You should be able to add more memory and disks as your applications grow.

ORACLE

Quiz

Which hardware component is the most frequent bottleneck for CPU-bound workloads?

a. Disks
b. Memory bandwidth
c. Network card
d. Storage area network (SAN)

Answer: b

Configuring Hardware

- Make sure it works.
  - Check for bad drivers if hardware is not working correctly.
- Check for proper throughput.
  - Check both random and sequential read/writes.
  - Verify that the disk cache is set to the proper mode for your system.
    - Benchmark your settings.
- Verify that the network interface is working properly.
  - Check to make sure that it is set to proper mode (that is, 1GB/full duplex).
  - Verify that CPU offloading is working.
  - Check the interrupt rates.
- Check for broken memory.

ORACLE

Configuring Hardware

Network interrupt rates

- Some drivers seem to have problems with buffering (taking an interrupt for each packet).

Broken memory

- Frequent source of MySQL “bugs”
- Test the memory with memtest86 or equivalent application if you are unsure of the status of your memory.

Quiz

Which of the following is a frequent source of MySQL bugs?

a. Broken memory
b. Disk cache set to wrong mode
c. Network card drivers
d. Other hardware drivers

Answer: a

Operating System Considerations

- MySQL supports a wide range of operating systems.
  - Windows, Linux, and Solaris are the top three.
  - Use operating systems for which Oracle delivers packages.
  - Verify that the vendor of an O/S can provide support.
    - Some non-Oracle O/S bugs cannot be fixed by Oracle support.
- Verify that the O/S has good thread support.
  - Ensure that the O/S contains kernel-level threads library for symmetric multiprocessing (SMP) support.
- Verify that the O/S allows you to address system memory.
- Verify that all your hardware is supported by the O/S.

Quiz

Which of the following operating systems are in the top three operating systems supported by MySQL? (Choose three.)

a. HP-UX
b. Linux
c. Solaris
d. Windows

Answer: b, c, d

Operating System Configuration

- Allow large process sizes.
- Allow a fairly large number of open files.
  - This is especially important for your MyISAM applications.
- When using a virtual machine (VM), ensure it is tuned well.
  - This is especially important to avoid swapping.
- Set proper I/O scheduling to ensure the best I/O performance.
- Consider using CPU binding (hard affinity) to prevent the scheduler from switching threads too often.
- Use large pages for MySQL processes if allowed by your O/S.

ORACLE

Operating System Configuration

Large process sizes

- The MySQL server is a single process. Allowing MySQL to use a large number of processes will provide benefits to performance.

Virtual machines

- It is also important that you size your MySQL buffers carefully when using a virtual machine (VM).

I/O scheduling

- The deadline elevator on Linux systems aggressively reorders requests to improve I/O performance.
  - The scheduler provides near real-time behavior and uses a round-robin policy to attempt to be fair among multiple I/O requests and to avoid process starvation.

CPU binding

- For more information about CPU affinity, see http://littledaemons.wordpress.com/2009/01/22/cpu-affinity-and-taskset.

Large pages

- Large pages are single-page table entries that cover large (up to multiple-megabyte) ranges of contiguous physical memory.
- After large pages are enabled on your O/S, enable the `--large-pages` MySQL server variable so that MySQL can take advantage of O/S large-page support.

Operating System Configuration

- For applications that use InnoDB, use direct I/O:
  - `O_DIRECT` on Linux systems; `directio` on Solaris systems
  - For logs and MyISAM applications, use I/O buffering.
- Increase the number of active commands for SCSI devices.
  - The default is often too low.
- Use large file system/block events.
  - Tables are typically large.
  - Use `notail` with ReiserFS.

Operating System Configuration (continued)

Direct I/O

- `O_DIRECT` attempts to minimize I/O cache effects to and from the application.

Active commands for SCSI devices

- The SCSI protocol allows multiple commands to be active on a logical unit number (LUN) at the same time.
  - A LUN is a unique identifier that is used on a SCSI bus to distinguish between devices that share the same bus.
- SCSI device drivers have a configurable parameter called the *LUN queue depth* that determines how many commands can be active at one time for a given LUN.

Large file system/block events

- ReiserFS is a general-purpose, journaled computer file system currently supported on Linux.

Quiz

For which type of application is it especially important to set your operating system to allow a fairly large number of open files?

a. C++
b. InnoDB
c. MyISAM
d. Web-based

Answer: c

Logging

- Use MyISAM tables with no indexes.
  - Provides fast logging and scans
  - Using the `ARCHIVE` storage engine can reduce the size of the footprint.
- Use `INSERT_DELAYED` to have live reporting.
  - When there are no holes, concurrent insert works as well.
  - It is possible to write requests to files and use a separate "feeder."
- When using indexes, keep their number small.
  - Keep tables small so that indexes fit in memory.
- Consider using multiple tables, partitioning, and/or fast data purging.

ORACLE

Logging

MyISAM tables

- Using MySQL to store logs provides an easy reporting tool using SQL.
- For example, the following SQL statement creates a report based on data in a log table:

```
SELECT AVG(request_time) FROM log WHERE request="search"
```

Quiz

Which storage engine is well suited for logging because it provides fast logging and scans?

a. `ARCHIVE`
b. InnoDB
c. `MEMORY`
d. MyISAM

Answer: d

Backup and Recovery

- Maintain your binary logs for the last two backups.
- Test your backups regularly to ensure that they actually restore valid data.
- Test how long a restoration of your data would take.
  - Textual backups take a long time to restore.
- Test how long roll-forward recovery takes.
  - `mysqlbinlog logfile015.bin --start-position=123 | mysql`
  - It may take up to several hours for each live hour.
    - Roll forward recovery is done by single thread .
  - Set `innodb_flush_log_at_trx_commit=0` for recovery.

Quiz

How long should you maintain your binary logs when used as a backup method?

a. One week
b. One month
c. Last two backups
d. Last five backups

Answer: d

Summary

In this lesson, you should have learned how to:

- Describe how your hardware and O/S can affect performance
- Demonstrate how to set up and evaluate logging to improve performance
- Describe best practices for backup and recovery

12

Conclusion

Course Goals: Review (1 of 3)

In this course, you should have learned how to:

- Define a tuning strategy for your organizational needs
- Explain how the MySQL architecture affects performance tuning
- Use diagnostic tools to evaluate the current performance conditions of the MySQL server
- Use tuning tools to improve the performance of the MySQL server
- Evaluate how the user base of your application interacts with the MySQL server, and implement strategies that work with your specific users' needs

ORACLE

Course Goals: Review (2 of 3)

In this course, you should have learned how to:

- Use the `INFORMATION_SCHEMA` database to assist with monitoring and evaluating the effectiveness of performance tuning on your MySQL server
- Describe how the relational database model affects database performance
- Describe the status variables that have direct and indirect effects on the performance of applications
- Adjust server configuration variables to improve the performance of applications
- Write queries that take advantage of the performance enhancements of the MySQL server

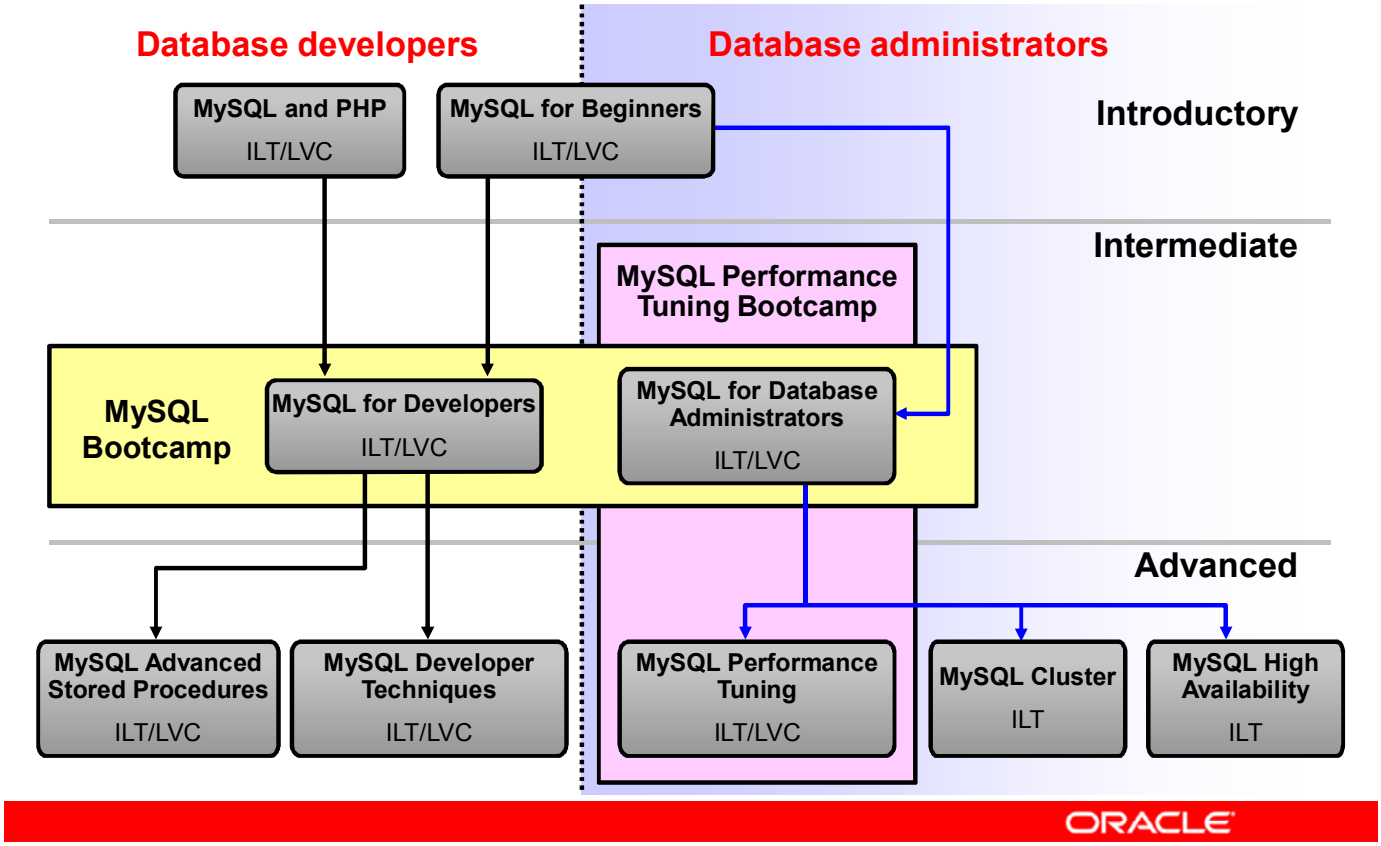
ORACLE

Course Goals: Review (3 of 3)

In this course, you should have learned how to:

- Describe how the storage engines have an effect on the performance of applications
- Describe how your current hardware and operating system affects the performance of the MySQL server
- Implement strategies for improving performance on data-loading operations

MySQL Curriculum Path



MySQL Resources

- MySQL training
  - http://www.oracle.com/education
- MySQL certification
  - http://www.oracle.com/certification
- Additional learning resources
  - http://msyql.com
    - Webinars and demos
    - Articles
    - White papers
- Developer Zone
  - http://dev.mysql.com/
    - Product downloads
    - Community communication

ORACLE

We Need Your Evaluation!

- Because we continually update our courses, your feedback is invaluable.
- Thank you for taking the time to give us your opinions!

Thank You!

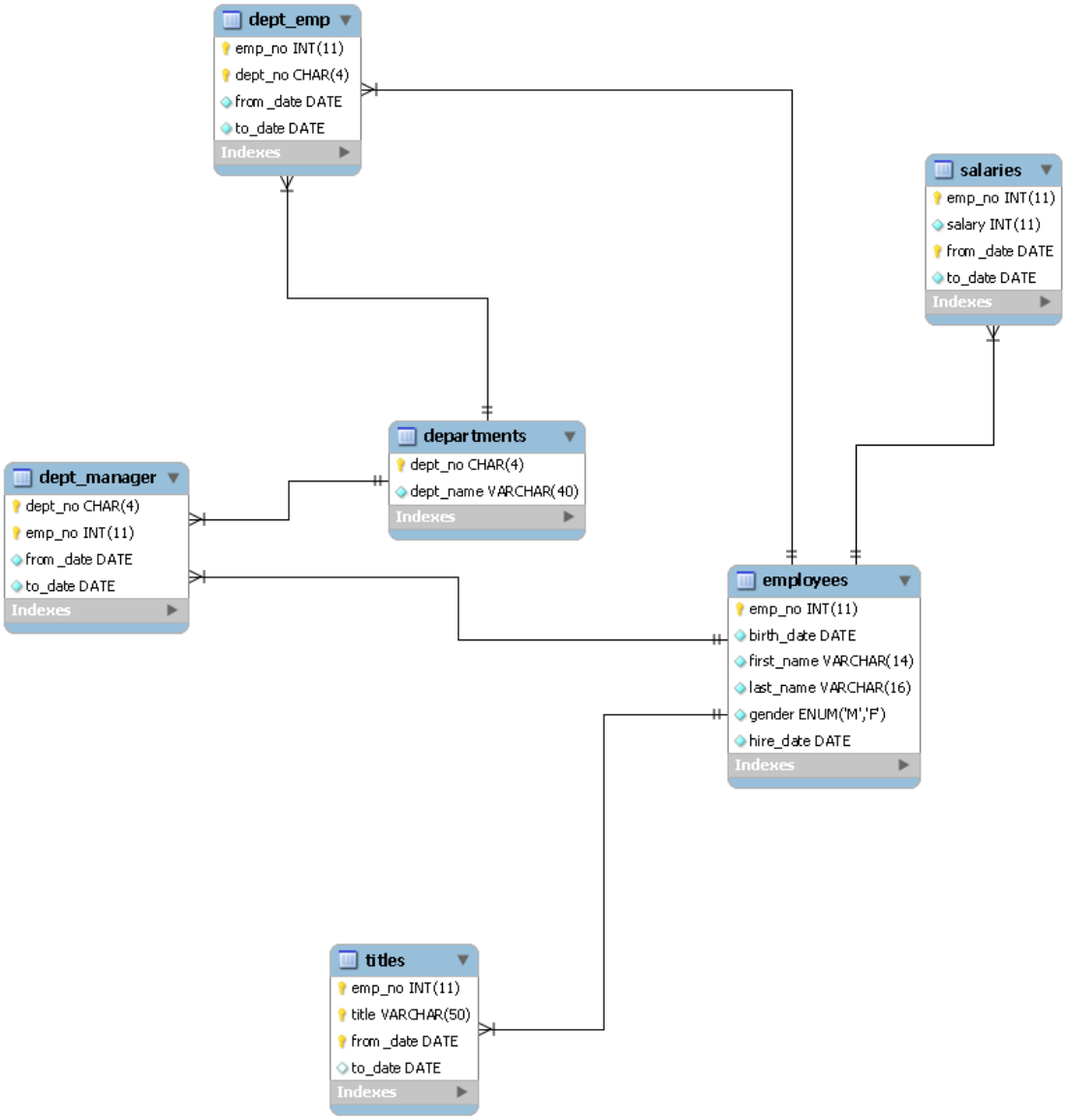
- Congratulations on completing this course!
- We appreciate your attendance and participation.
- For training information and to contact us, please see the Oracle University website at http://www.oracle.com/us/education.

Q&A Session

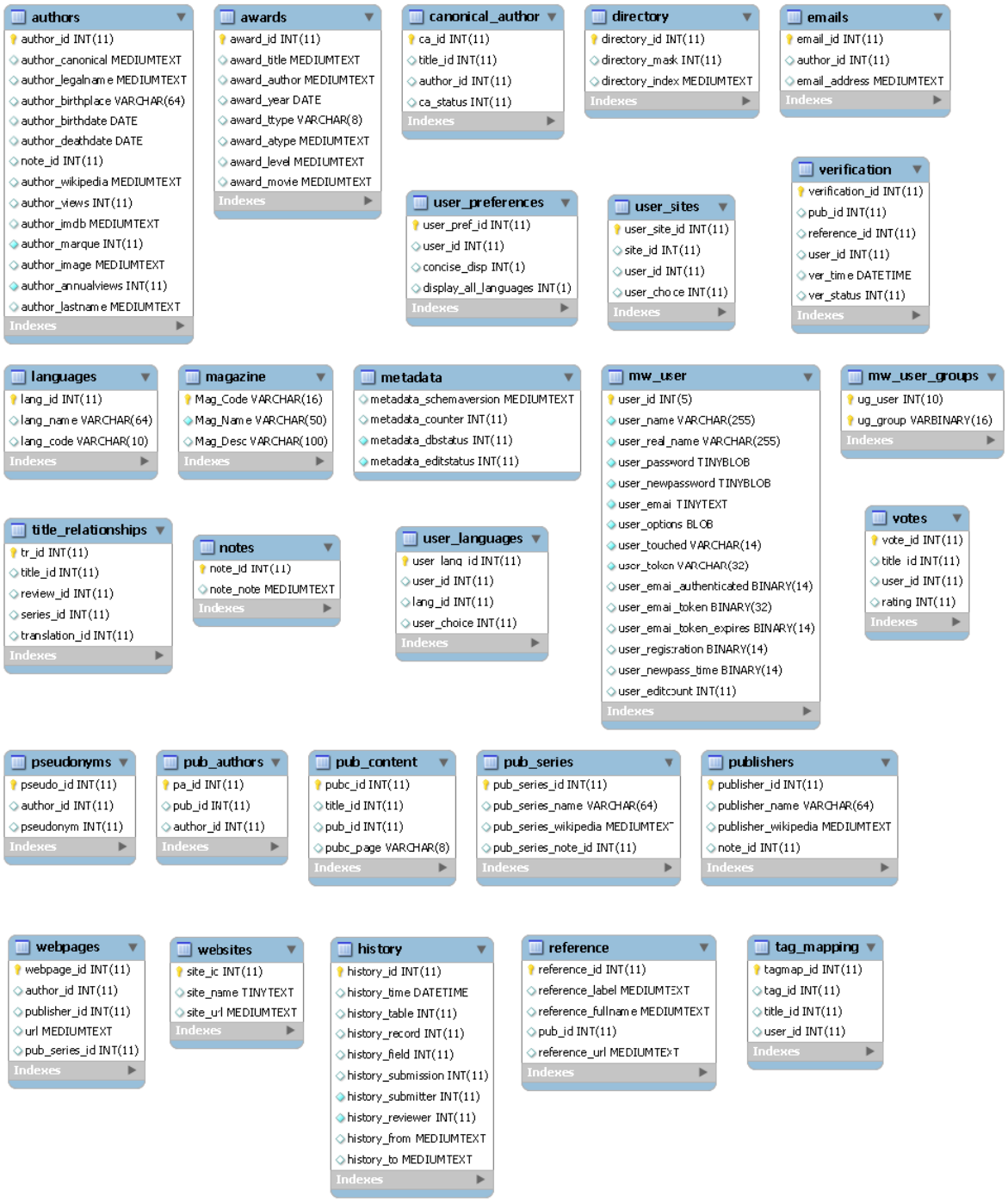
- Questions and answers
- Questions after class
  - Get answers from our online Reference Manual.
  - http://dev.mysql.com/doc/refman/5.5/en/faqs.html
- Example databases
  - Download the `world_innodb` database (as well as others) from our website.
  - http://dev.mysql.com/doc/index-other.html (under “Example Databases”)

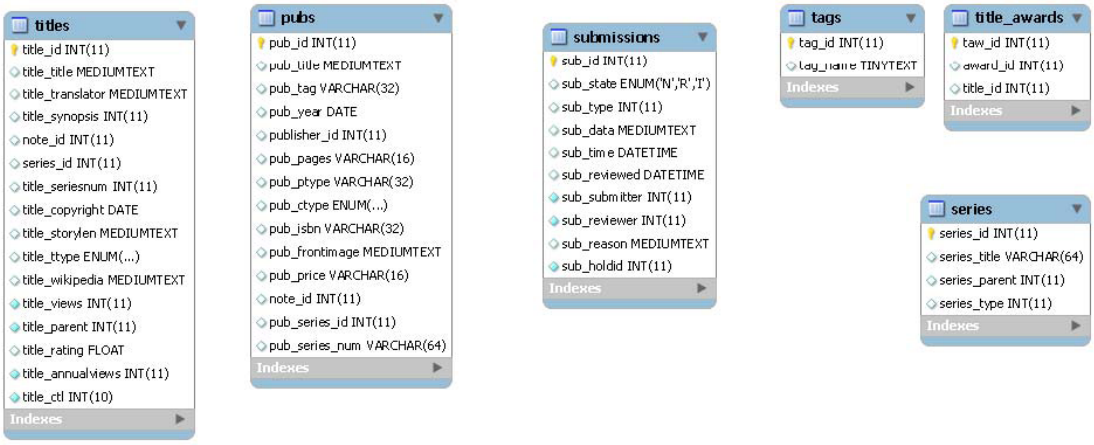
Example Schema Data Models

Employees Schema Data Model

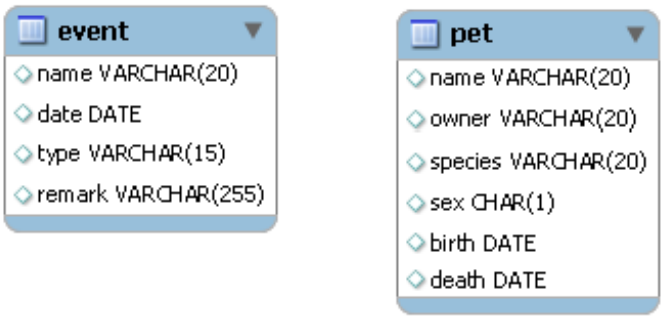


Internet Speculative Fiction Database Schema Data Model

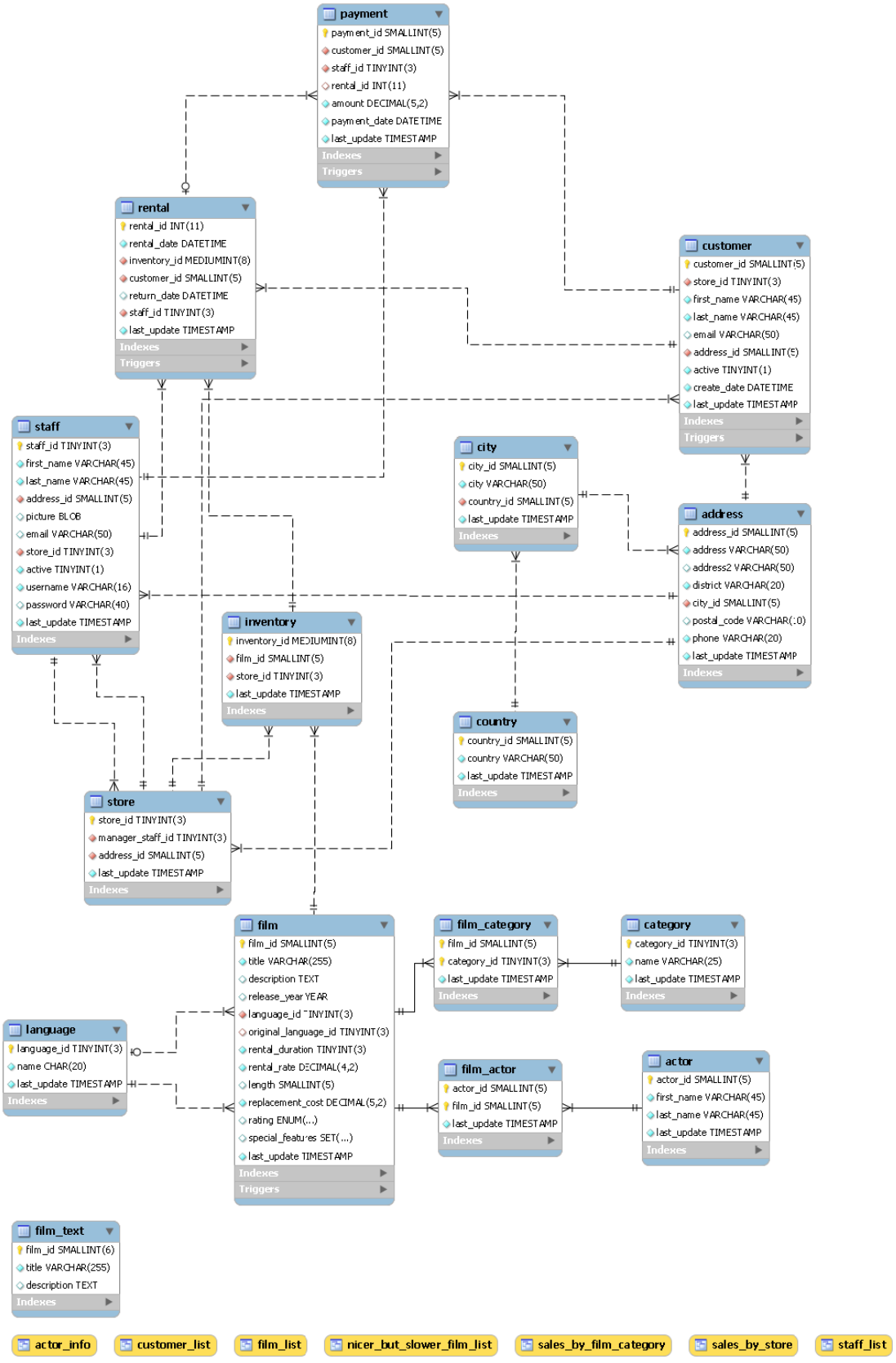




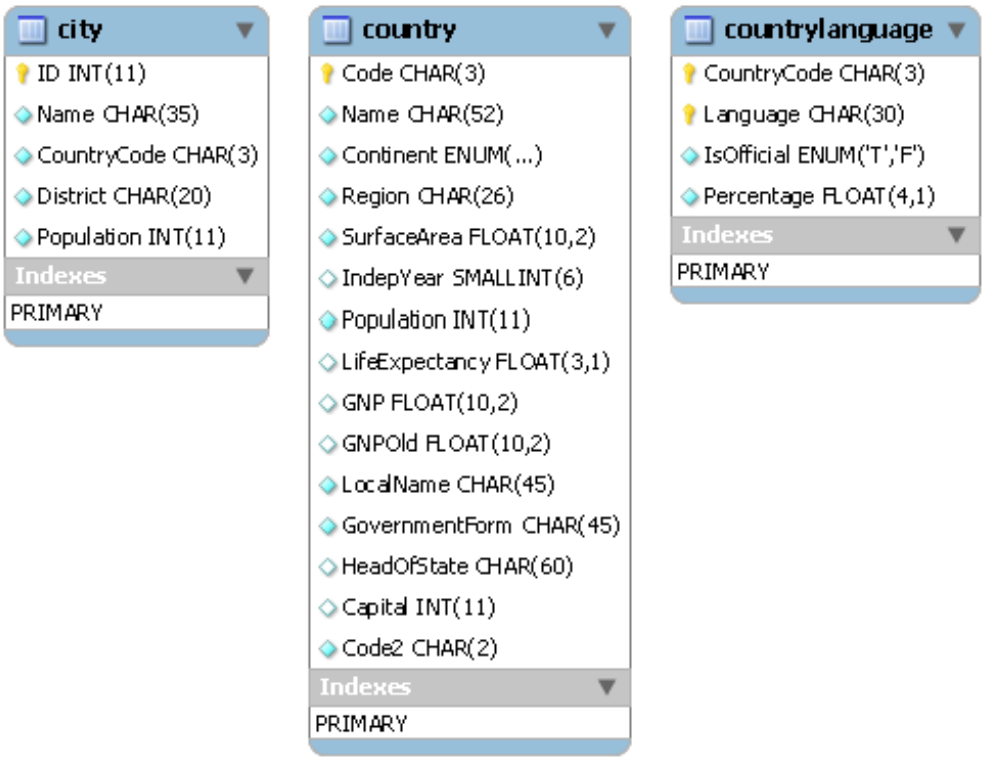
Menagerie Schema Data Model



Sakila Schema Data Model



World Schema Data Model



Lab Scripts

Lesson 1: Introduction

No activities for this lesson.

Lesson 2: Performance Tuning Basics

No activities for this lesson

Lesson 3: Performance Tuning Tools

Practice 3-1 MySQL Monitoring Tools

```
su -
rm -rf /etc/init.d/mysql-monitor-server
ln -s /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh \
/etc/init.d/mysql-monitor-server
/etc/init.d/mysql-monitor-server stop
/etc/init.d/mysql-monitor-agent stop
exit
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads%'"
mysql -uroot -poracle -e"SHOW TABLES FROM INFORMATION_SCHEMA LIKE 'S%'"
mysql -uroot -poracle -e"DESC INFORMATION_SCHEMA.SESSION_STATUS"
mysql -uroot -poracle -e"SELECT * FROM
INFORMATION_SCHEMA.SESSION_STATUS WHERE VARIABLE_NAME LIKE 'Thread%'"
su -
/etc/init.d/mysql stop
/etc/init.d/mysql start --performance_schema
exit
mysql -uroot -poracle -e"SHOW TABLES FROM performance_schema"
mysql -uroot -poracle -e"SELECT * FROM performance_schema.threads"
mysql -uroot -poracle < /stage/scripts/10min.sql
### Second Terminal Window
mysql -uroot -poracle -e"SHOW OPEN TABLES WHERE In_use > 0"
mysql -uroot -poracle -e"SHOW PROCESSLIST\G"
mysqladmin -uroot -poracle status
mysqladmin -uroot -poracle kill 5
### First Terminal Window
su -
/etc/init.d/mysql-monitor-server start
/etc/init.d/mysql-monitor-agent start
/etc/init.d/mysql-monitor-agent stop
/etc/init.d/mysql-monitor-server stop
/etc/init.d/mysql restart
exit
```

Practice 3-2 Open Source Community Tools

```
su -
/etc/init.d/mysql stop
/etc/init.d/apachectl stop
/apps/ADMIN/dim_STAT-Server start
/etc/STATsrv/STAT-service start
/etc/STATsrv/STAT-service stop
/apps/ADMIN/dim_STAT-Server stop
/etc/init.d/apachectl start
/etc/init.d/mysql start
```

```
/stage/mysqlresources -u root -p oracle
exit
ls /stage/maatkit-7041/bin
mk-find -uroot -poracle --engine InnoDB
mk-find -uroot -poracle --tablesize +50M
mk-variable-advisor -uroot -poracle localhost
su -
/etc/init.d/mysql restart
cd /stage/mysqlreport-3.5
mysqlreport --user root --password oracle > /tmp/mysqlreport_class1.txt
exit
more /tmp/mysqlreport_class1.txt
mysql -uroot -poracle < /stage/scripts/Qcache-queries.sql
su -
cd /stage/mysqlreport-3.5
mysqlreport --user root --password oracle > /tmp/mysqlreport_class2.txt
exit
more /tmp/mysqlreport_class2.txt
mytop --user root --password oracle -d world
```

Practice 3-3 Benchmark Tools

```
su -
/etc/init.d/mysql restart
cd /usr/bin/sql-bench
perl run-all-tests --small-test --create-options=ENGINE=INNODB --
user="root" --password="oracle" | tee /tmp/sql-bench-run-all-tests.txt
exit
ls /usr/bin/sql-bench/output/
more /usr/bin/sql-bench/output/select-mysql-Linux_2.6.18_164.el5_i686
more /tmp/sql-bench-run-all-tests.txt
mysql -uroot -poracle -e"DROP DATABASE IF EXISTS mysqlslap;CREATE
DATABASE mysqlslap;"
su -
mysqlslap --user=root --password=oracle \
--query="/stage/scripts/dept_employees.sql" --iterations=10 \
--concurrency=10
exit
sysbench --test=oltp --mysql-table-engine=myisam \
--oltp-table-size=50000 --mysql-socket=/var/lib/mysql/mysql.sock \
--mysql-user=root --mysql-password=oracle --mysql-db=test prepare
sysbench --test=oltp --oltp-table-size=50000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
mysql -uroot -poracle -e"DROP DATABASE test;CREATE DATABASE test;"
```

Lesson 4: MySQL Server Tuning

Practice 4-1 Effects of Thread Caching

```
su -
/etc/init.d/mysql restart
exit
sysbench --test=oltp --mysql-table-engine=myisam \
--oltp-table-size=1000000 --mysql-socket=/var/lib/mysql/mysql.sock \
--mysql-user=root --mysql-password=oracle --mysql-db=test prepare
mysql -uroot -poracle -e"SET GLOBAL thread_cache_size=0; SET GLOBAL
max_connections=300;"
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads_created'"
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Connections'"
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads_created'"
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Connections'"
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads_created'"
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Connections'"
su -
/etc/init.d/mysql restart
exit
mysql -uroot -poracle -e"SET GLOBAL thread_cache_size=8; SET GLOBAL
max_connections=300;"
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads_created'"
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Connections'"
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads_created'"
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Connections'"
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
```

```
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads_created'"
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Connections'"
su -
/etc/init.d/mysql restart
exit
mysql -uroot -poracle -e"SET GLOBAL thread_cache_size=16; SET GLOBAL
max_connections=300;"
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads_created'"
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Connections'"
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads_created'"
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Connections'"
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=128 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Threads_created'"
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Connections'"
```

Practice 4-2 Table Caching

```
su -
/etc/init.d/mysql stop
/etc/init.d/mysql start --table-open-cache=1 --query-cache-size=0
exit
cd /stage/scripts
./PT_Stress.php -h
./PT_Stress.php -u root -p oracle -s "dept_employees_large.sql" \
-i 20 -c 5 -t -v "open%table%"
su -
/etc/init.d/mysql stop
/etc/init.d/mysql start --table-open-cache=8 --query-cache-size=0
exit
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "dept_employees_large.sql" \
-i 20 -c 5 -t -v "open%table%"
su -
/etc/init.d/mysql stop
/etc/init.d/mysql start --table-open-cache=16 --query-cache-size=0
exit
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "dept_employees_large.sql" \
-i 20 -c 5 -t -v "open%table%"
```

```
su -
/etc/init.d/mysql stop
/etc/init.d/mysql start --table-open-cache=32 --query-cache-size=0
exit
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "dept_employees_large.sql" \
-i 20 -c 5 -t -v "open%table%"
su -
/etc/init.d/mysql stop
/etc/init.d/mysql start --table-open-cache=48 --query-cache-size=0
exit
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "dept_employees_large.sql" \
-i 20 -c 5 -t -v "open%table%"
su -
/etc/init.d/mysql stop
/etc/init.d/mysql start --table-open-cache=64 --query-cache-size=0
exit
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "dept_employees_large.sql" \
-i 20 -c 5 -t -v "open%table%"
```

Practice 4-3 Setting `open_files_limit`

```
cat /proc/meminfo
su -
sysctl fs.file-max
exit
mysql -uroot -poracle -e"SELECT @@global.open_files_limit"
su -
vi /etc/my.cnf
### open_files_limit=453120
/etc/init.d/mysql restart
exit
mysql -uroot -poracle -e"SELECT @@global.open_files_limit"
```

Practice 4-4 Setting `max_connections`

```
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'max_connect%'";
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=200 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Max_used_connections'";
mysql -uroot -poracle -e"SET GLOBAL max_connections = 225";
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=200 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Max_used_connections'";
```

```
su -
/etc/init.d/mysql restart
exit
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=200 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Max_used_connections'";
su -
vi /etc/my.cnf
### max_connections = 225
/etc/init.d/mysql restart
exit
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=1000 --num-threads=200 run
mysql -uroot -poracle -e"SHOW STATUS LIKE 'Max_used_connections'";
```

Practice 4-5 Evaluating the Effects of Numerous Connections

```
su -
/etc/init.d/mysql restart
exit
mysql -uroot -poracle -e"SET GLOBAL max_connections = 1024";
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=100000 --num-threads=8 \
run | grep "total time"
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=100000 --num-threads=32 \
run | grep "total time"
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=100000 --num-threads=64 \
run | grep "total time"
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
```

```
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=100000 --num-threads=128 \
run | grep "total time"
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=100000 --num-threads=256 \
run | grep "total time"
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=100000 --num-threads=512 \
run | grep "total time"
### Second Terminal Window
top -n 8 | grep mysqld
sysbench --test=oltp --oltp-table-size=1000000 \
--mysql-socket=/var/lib/mysql/mysql.sock --oltp-test-mode=simple \
--oltp-reconnect-mode=query --mysql-user=root --mysql-password=oracle \
--mysql-db=test --max-requests=100000 --num-threads=1000 \
run | grep "total time"
### Second Terminal Window
top -n 8 | grep mysqld
```

Practice 4-6 Total Thread Memory Usage

```
su -
vi /etc/my.cnf
### read_buffer_size:     2M
### read_rnd_buffer_size: 8M
### sort_buffer_size:     2M
### max_connections:      225
exit
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'join_buffer_size'";
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'thread_stack'";
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'max_heap_table_size'";
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'tmp_table_size'";
```

Practice 4-7 Sort Queries

```
su -
/etc/init.d/mysql restart
exit
```

```
mysql -uroot -poracle -e"SET GLOBAL sort_buffer_size=32768;SET GLOBAL
query_cache_type=0"
cd /stage/scripts
./PT_Stress.php -u root -p oracle -c 20 -s "sort.read" \
-i 1 -v "Sort%" -t
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL sort_buffer_size=524288"
./PT_Stress.php -u root -p oracle -c 20 -s "sort.read" \
-i 1 -v "Sort%" -t
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL sort_buffer_size=1048576"
./PT_Stress.php -u root -p oracle -c 20 -s "sort.read" \
-i 1 -v "Sort%" -t
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL sort_buffer_size=4194304"
./PT_Stress.php -u root -p oracle -c 20 -s "sort.read" \
-i 1 -v "Sort%" -t
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL sort_buffer_size=8388608"
./PT_Stress.php -u root -p oracle -c 20 -s "sort.read" \
-i 1 -v "Sort%" -t
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL sort_buffer_size=16777216"
./PT_Stress.php -u root -p oracle -c 20 -s "sort.read" \
-i 1 -v "Sort%" -t
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL sort_buffer_size=33554432"
./PT_Stress.php -u root -p oracle -c 20 -s "sort.read" \
-i 1 -v "Sort%" -t
### Second Terminal Window
top -n 8 | grep mysqld
```

Lesson 5: MySQL Query Cache

Practice 5-1 Query Cache

```
su -
/etc/init.d/mysql restart
exit
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'have%'"
mysql -uroot -poracle -e"SET GLOBAL query_cache_type = 1"
mysql -uroot -poracle -e"SET GLOBAL query_cache_size = 0"
./PT_Stress.php -u root -p oracle -c 10 -s "qcache.read" \
-i 5 -v "QCache_hits,Com_select" -t
### Second Terminal Window
top | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL query_cache_size = 524288"
./PT_Stress.php -u root -p oracle -c 10 -s "qcache.read" \
-i 5 -v "QCache_hits,Com_select" -t
### Second Terminal Window
top | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL query_cache_size = 1048576"
./PT_Stress.php -u root -p oracle -c 10 -s "qcache.read" \
-i 5 -v "QCache_hits,Com_select" -t
### Second Terminal Window
top | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL query_cache_size = 4194304"
./PT_Stress.php -u root -p oracle -c 10 -s "qcache.read" \
-i 5 -v "QCache_hits,Com_select" -t
### Second Terminal Window
top | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL query_cache_size = 16777216"
./PT_Stress.php -u root -p oracle -c 10 -s "qcache.read" \
-i 5 -v "QCache_hits,Com_select" -t
### Second Terminal Window
top | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL query_cache_size = 67108864"
./PT_Stress.php -u root -p oracle -c 10 -s "qcache.read" \
-i 5 -v "QCache_hits,Com_select" -t
### Second Terminal Window
top | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL query_cache_size = 134217728"
./PT_Stress.php -u root -p oracle -c 10 -s "qcache.read" \
-i 5 -v "QCache_hits,Com_select" -t
```

```
### Second Terminal Window
top | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL query_cache_size = 268435456"
./PT_Stress.php -u root -p oracle -c 10 -s "qcache.read" \
-i 5 -v "QCache_hits,Com_select" -t
### Second Terminal Window
top | grep mysqld
### First Terminal Window
mysql -uroot -poracle -e"SET GLOBAL query_cache_size = 536870912"
./PT_Stress.php -u root -p oracle -c 10 -s "qcache.read" \
-i 5 -v "QCache_hits,Com_select" -t
### Second Terminal Window
top | grep mysqld
su -
vi /etc/init.d/my.cnf
### query_cache_type = 0
/etc/init.d/mysql restart
exit
```

Lesson 6: InnoDB

Practice 6-1 Committing Transactions

```
mysql -uroot -poracle
USE many_tables;
SELECT ROUTINE_NAME, ROUTINE_DEFINITION FROM
INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME LIKE 'innodb_inserts'\G
CALL innodb_inserts(100000,1);
CALL innodb_inserts(100000,5);
CALL innodb_inserts(100000,10);
CALL innodb_inserts(100000,100);
CALL innodb_inserts(100000,1000);
CALL innodb_inserts(100000,5000);
CALL innodb_inserts(100000,10000);
CALL innodb_inserts(100000,50000);
CALL innodb_inserts(100000,100000);
```

Practice 6-2 `SHOW ENGINE INNODB STATUS`

```
su -
/etc/init.d/mysql restart
exit
mysql -uroot -poracle
USE many_tables;
SELECT ROUTINE_NAME, ROUTINE_DEFINITION FROM
INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME LIKE
'create_city_huge'\G
SELECT ROUTINE_NAME, ROUTINE_DEFINITION FROM
INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME LIKE
'add_data_to_city_huge'\G
CALL create_city_huge(100);
ALTER TABLE city_huge ENGINE=INNODB;
### Second Terminal Window
su -
mysqlslap -uroot -poracle -q /stage/scripts/innodb_query.sql \
--create-schema=many_tables -i 20 -c 20
exit
### First Terminal Window
SHOW ENGINE INNODB STATUS\G
```

Practice 6-3 InnoDB Monitors

```
ls /var/lib/mysql/*.err


su -
## Modify the information below for the machine you are using
## cp /var/lib/mysql/EDTDR14P0.err /var/lib/mysqlEDTDR14P0.err-bu1
## rm -rf /var/lib/mysql/EDTDR14P0.err
/etc/init.d/mysql restart
```

```
exit
### Second Terminal Window
mysql -uroot -poracle
USE many_tables;
CREATE TABLE innodb_monitor (A INT) ENGINE=INNODB;
### Wait one minute, the SLEEP function can be used to keep the script
### from moving on until one minute has passed
SELECT SLEEP(60);
DROP TABLE innodb_monitor;
### First Terminal Window
su -
## Modify the information below for the machine you are using
## more /var/lib/mysql/EDTDR14P0.err
## cp /var/lib/mysql/EDTDR14P0.err /var/lib/mysqlEDTDR14P0.err-bu2
## rm -rf /var/lib/mysql/EDTDR14P0.err
/etc/init.d/mysql restart
exit
### Second Terminal Window
CREATE TABLE innodb_tablespace_monitor (A INT) ENGINE=INNODB;
### Wait one minute, the SLEEP function can be used to keep the script
### from moving on until one minute has passed
SELECT SLEEP(60);
DROP TABLE innodb_tablespace_monitor;
### First Terminal Window
su -
## Modify the information below for the machine you are using
## more /var/lib/mysql/EDTDR14P0.err
## cp /var/lib/mysql/EDTDR14P0.err /var/lib/mysqlEDTDR14P0.err-bu3
## rm -rf /var/lib/mysql/EDTDR14P0.err
/etc/init.d/mysql restart
exit
### Second Terminal Window
CREATE TABLE innodb_table_monitor (A INT) ENGINE=INNODB;
### Wait one minute, the SLEEP function can be used to keep the script
### from moving on until one minute has passed
SELECT SLEEP(60);
DROP TABLE innodb_table_monitor;
### First Terminal Window
su -
## Modify the information below for the machine you are using
## more /var/lib/mysql/EDTDR14P0.err
exit
### Second Terminal Window
exit;
```

Practice 6-4 InnoDB Settings

```
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'innodb%'"
su -
mysqlslap -uroot -poracle -q /stage/scripts/innodb_query.sql \
--create-schema=many_tables -i 4 -c 10
exit
mysql -uroot -poracle -e"SET GLOBAL innodb_flush_log_at_trx_commit=2"
su -
mysqlslap -uroot -poracle -q /stage/scripts/innodb_query.sql \
--create-schema=many_tables -i 4 -c 10
exit
mysql -uroot -poracle -e"SET GLOBAL innodb_flush_log_at_trx_commit=0"
su -
mysqlslap -uroot -poracle -q /stage/scripts/innodb_query.sql \
--create-schema=many_tables -i 4 -c 10
/etc/init.d/mysql restart --innodb_buffer_pool_size=0
mysqlslap -uroot -poracle -q /stage/scripts/innodb_buffer.sql \
--create-schema=many_tables -i 5 -c 1
exit
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
su -
/etc/init.d/mysql restart --innodb_buffer_pool_size=12M
mysqlslap -uroot -poracle -q /stage/scripts/innodb_buffer.sql \
--create-schema=many_tables -i 5 -c 1
exit
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
su -
/etc/init.d/mysql restart --innodb_buffer_pool_size=24M
mysqlslap -uroot -poracle -q /stage/scripts/innodb_buffer.sql \
--create-schema=many_tables -i 5 -c 1
exit
### Second Terminal Window
top -n 8 | grep mysqld
### First Terminal Window
su -
/etc/init.d/mysql restart --innodb_buffer_pool_size=36M
mysqlslap -uroot -poracle -q /stage/scripts/innodb_buffer.sql \
--create-schema=many_tables -i 5 -c 1
exit
### Second Terminal Window
top -n 8 | grep mysqld
```

Lesson 7: MyISAM

Practice 7-1 Optimizing MyISAM

```
mysql -uroot -poracle -e"USE many_tables;CALL create_city_huge(0);\
ALTER TABLE city_huge ENGINE=MYISAM;\
ALTER TABLE city_huge DISABLE KEYS;\
SOURCE /stage/scripts/city_huge.sql;ALTER TABLE city_huge ENABLE KEYS;"
mysql -uroot -poracle -e"SELECT COUNT(*) FROM many_tables.city_huge;"
su -
ls -l /var/lib/mysql/many_tables/city_huge.*
exit
mysql -uroot -poracle -e"USE many_tables;\
SOURCE /stage/scripts/delete_city_huge.sql;"
mysql -uroot -poracle -e"SELECT COUNT(*) FROM many_tables.city_huge;"
su -
ls -l /var/lib/mysql/many_tables/city_huge.*
exit
mysql -uroot -poracle -e"OPTIMIZE TABLES many_tables.city_huge;"
su -
ls -l /var/lib/mysql/many_tables/city_huge.*
exit
```

Practice 7-2 MyISAM Table Locks

```
mysql -uroot -poracle -e"USE many_tables;TRUNCATE city_huge;\
ALTER TABLE city_huge DISABLE KEYS;SOURCE
/stage/scripts/city_huge.sql;\ ALTER TABLE city_huge ENABLE KEYS;"
# Note: For the next few steps, you need to have four separate terminal
windows opened.  The first terminal window is running the mysql client.
The second terminal window is running delete operations against the
MySQL server.  The third terminal window is running insert operations
against the MySQL server. The fourth terminal window is running the
select operations.
### Terminal 1
mysql -uroot -poracle
FLUSH STATUS;
SHOW FULL PROCESSLIST\G
SHOW GLOBAL STATUS LIKE 'Table_locks%';
### Terminal 2
time mysql -uroot -poracle -e"USE many_tables;\
SOURCE /stage/scripts/delete_city_huge_32K.sql;"
### Terminal 3
time mysql -uroot -poracle -e" USE many_tables;\
SOURCE /stage/scripts/insert_city_huge_100K.sql;"
### Terminal 4
cd /stage/scripts
./PT_Stress.php -u root -p oracle -t -q -s "Q:SELECT * FROM
many_tables.city_huge WHERE ID={RANDOM}" -i 100000 -r 2000000
### Terminal 1
SHOW FULL PROCESSLIST\G
```

```
SHOW GLOBAL STATUS LIKE 'Table_locks%';
FLUSH STATUS;
### Terminal 2
mysql -uroot -poracle -e"USE many_tables;TRUNCATE city_huge;\
ALTER TABLE city_huge DISABLE KEYS;SOURCE
/stage/scripts/city_huge.sql;\ ALTER TABLE city_huge ENABLE KEYS;"
### Terminal 2
time mysql -uroot -poracle -e"USE many_tables;\
SOURCE /stage/scripts/delete_city_huge_8K_1.sql;\
SELECT SLEEP(30);SOURCE /stage/scripts/delete_city_huge_8K_2.sql;\
SELECT SLEEP(30);SOURCE /stage/scripts/delete_city_huge_8K_3.sql;\
SELECT SLEEP(30);SOURCE /stage/scripts/delete_city_huge_8K_4.sql;"
### Terminal 3
time mysql -uroot -poracle -e"USE many_tables;\
SOURCE /stage/scripts/insert_city_huge_100K.sql;"
### Terminal 4
cd /stage/scripts
./PT_Stress.php -u root -p oracle -t -q -s "Q:SELECT * FROM
many_tables.city_huge WHERE ID={RANDOM}" -i 100000 -r 2000000
### Terminal 1
SHOW GLOBAL STATUS LIKE 'Table_locks%';
FLUSH STATUS;
### Terminal 2
mysql -uroot -poracle -e"USE many_tables;TRUNCATE city_huge;\
ALTER TABLE city_huge DISABLE KEYS;SOURCE
/stage/scripts/city_huge.sql;\ ALTER TABLE city_huge ENABLE KEYS;"
### Terminal 3
time mysql -uroot -poracle -e"USE many_tables;\
SOURCE /stage/scripts/insert_city_huge_100K.sql;"
### Terminal 4
cd /stage/scripts
./PT_Stress.php -u root -p oracle -t -q -s "Q:SELECT * FROM
many_tables.city_huge WHERE ID={RANDOM}" -i 100000 -r 2000000
### Terminal 1
SHOW GLOBAL STATUS LIKE 'Table_locks%';
```


Practice 7-3 Key Cache Effectiveness

```
su -
/etc/init.d/mysql restart
exit
mysql -uroot -poracle -e"SET GLOBAL key_buffer_size=16*1024;"
mysql -uroot -poracle -e"SHOW GLOBAL VARIABLES \
LIKE 'key_cache_block_size';"
mysql -uroot -poracle -e"SHOW GLOBAL STATUS LIKE 'Key%';"
mysql -uroot -poracle -e"SELECT Name FROM world.Country \
WHERE Code = 'SWE';"
mysql -uroot -poracle -e"SHOW GLOBAL STATUS LIKE 'Key%';"
```

```
mysql -uroot -poracle -e"SELECT Name FROM world.Country \
WHERE Code = 'SWZ';"
mysql -uroot -poracle -e"SHOW GLOBAL STATUS LIKE 'Key%';"
mysql -uroot -poracle -e"SELECT Name FROM world.Country \
WHERE Code = 'FIN';"
mysql -uroot -poracle -e"SHOW GLOBAL STATUS LIKE 'Key%';"
```

Practice 7-4 Full-Text Indexing

```
mysql -uroot -poracle -e"USE many_tables;\
DROP TABLE IF EXISTS city_fulltext;\
CREATE TABLE city_fulltext LIKE city_huge;\
ALTER TABLE city_fulltext DROP INDEX CountryCode,\
DROP INDEX Name, DROP INDEX Population;\
ALTER TABLE city_fulltext ADD FULLTEXT(Name,District);\
INSERT INTO city_fulltext SELECT * FROM city_huge;"
mysql -uroot -poracle -e"SELECT COUNT(*) FROM \
many_tables.city_fulltext WHERE Name LIKE 'San %';"
mysql -uroot -poracle -e"SELECT COUNT(*) FROM \
many_tables.city_fulltext WHERE MATCH(Name,District) AGAINST('San');"
su -
/etc/init.d/my.cnf
### ft_min_word_len = 3
/etc/init.d/mysql restart
exit
mysql -uroot -poracle -e"USE many_tables;\
DROP TABLE IF EXISTS city_fulltext;\
CREATE TABLE city_fulltext LIKE city_huge;\
ALTER TABLE city_fulltext DROP INDEX CountryCode,\
DROP INDEX Name, DROP INDEX Population;\
ALTER TABLE city_fulltext ADD FULLTEXT(Name,District);\
INSERT INTO city_fulltext SELECT * FROM city_huge;"
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "Q:SELECT * FROM
many_tables.city_fulltext WHERE MATCH(Name,District) AGAINST ('San')
LIMIT 1000" -t -i 100
./PT_Stress.php -u root -p oracle -s "Q:SELECT * FROM
many_tables.city_fulltext WHERE MATCH(Name,District) AGAINST ('San' IN
BOOLEAN MODE) LIMIT 1000" -t -i 100
./PT_Stress.php -u root -p oracle -s "Q:SELECT COUNT(*) FROM
many_tables.city_fulltext WHERE MATCH(Name,District) AGAINST ('San')" \
 -t -i 100
./PT_Stress.php -u root -p oracle -s "Q:SELECT * FROM
many_tables.city_fulltext WHERE MATCH(Name,District) AGAINST ('San' IN
BOOLEAN MODE) LIMIT 1000" -t -i 100
```

Lesson 8: Other MySQL Storage Engines and Issues

Practice 8-1 Large Objects

```
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "Q:SELECT
picture FROM many_tables.BLOB_InnoDB" -t
./PT_Stress.php -u root -p oracle -s "Q:SELECT
picture FROM many_tables.BLOB_MyISAM" -t
./PT_Stress.php -u root -p oracle -s "Q:SELECT
last_name FROM many_tables.BLOB_InnoDB" -i 10 -c 10 -t
./PT_Stress.php -u root -p oracle -s "Q:SELECT
last_name FROM many_tables.BLOB_MyISAM" -i 10 -c 10 -t
```

Practice 8-2 `MEMORY` Storage Engine

```
mysql -uroot -poracle -e"SET GLOBAL max_heap_table_size = 100000 *
1024"
time mysql -uroot -poracle -e"USE many_tables;TRUNCATE city_huge;\
ALTER TABLE city_huge ENGINE=MYISAM;ALTER TABLE city_huge DISABLE
KEYS;\
SOURCE /stage/scripts/city_huge.sql;ALTER TABLE city_huge ENABLE KEYS;"
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "Q:SELECT id FROM
many_tables.city_huge WHERE
ID={RANDOM};" -r 2000000 -t -i 100000 -c 10 -q
### Terminal 2
top | grep mysqld
### Terminal 1
time mysql -uroot -poracle -e"USE many_tables;TRUNCATE city_huge;\
ALTER TABLE city_huge ENGINE=MEMORY;\
ALTER TABLE city_huge DISABLE KEYS;\
SOURCE  /stage/scripts/city_huge.sql;ALTER TABLE city_huge ENABLE
KEYS;"
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "Q:SELECT id FROM
many_tables.city_huge WHERE
ID={RANDOM};" -r 2000000 -t -i 100000 -c 10 -q
### Terminal 2
top | grep mysqld
```

Lesson 9: Schema Design and Performance

Practice 9-1 Schema Design

```
mysql -uroot -poracle -e"SHOW CREATE TABLE world_NonNorm.world_all\G"
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "Q:SELECT DISTINCT Country_Name,
GovernmentForm FROM world_NonNorm.world_all WHERE Country_Population <
{RANDOM}" -r 1500000000 -t -i 100 -c 5 -q
./PT_Stress.php -u root -p oracle -s "Q:SELECT Name, GovernmentForm
FROM world.Country WHERE Population
< {RANDOM}" -r 1500000000 -t -i 100 -c 5 -q
./PT_Stress.php -u root -p oracle -s "Q:SELECT City_Name, Country_Name
FROM world_NonNorm.world_all WHERE Percentage
< {RANDOM}" -r 101 -t -i 100 -c 2 -q
./PT_Stress.php -u root -p oracle -s "Q:SELECT City.Name, Country.Name
FROM world.City, world.Country, world.CountryLanguage WHERE
City.CountryCode = Country.Code AND Country.Code =
CountryLanguage.CountryCode AND Percentage
< {RANDOM}" -r 101 -t -i 100 -c 2 -q
```

Practice 9-2 Data Types

```
su -
ls -alt /var/lib/mysql/isfdb/authors*
exit
mysql -uroot -poracle -e"ALTER TABLE isfdb.authors MODIFY \
author_birthdate DATETIME DEFAULT NULL"
su -
ls -alt /var/lib/mysql/isfdb/authors*
exit
mysql -uroot -poracle -e"SHOW CREATE TABLE isfdb.history\G"
### Terminal 2
mysql -uroot -poracle -e"SELECT MAX(LENGTH(history_from)) \
FROM isfdb.history"
mysql -uroot -poracle -e"SELECT MAX(LENGTH(history_to)) \
FROM isfdb.history"
mysql -uroot -poracle -e"USE isfdb;CREATE TABLE history_char \
LIKE history;\
ALTER TABLE history_char MODIFY history_from CHAR(199);\
ALTER TABLE history_char MODIFY history_to CHAR(226);\
INSERT INTO history_char SELECT * FROM history;"
mysql -uroot -poracle -e"USE isfdb;CREATE TABLE history_varchar \
LIKE history;\
ALTER TABLE history_varchar MODIFY history_from VARCHAR(199);\
ALTER TABLE history_varchar MODIFY history_to VARCHAR(226);\
INSERT INTO history_varchar SELECT * FROM history;"
mysql -uroot -poracle -e"USE isfdb;CREATE TABLE history_mixed \
LIKE history;\
ALTER TABLE history_mixed MODIFY history_from CHAR(199);\
ALTER TABLE history_mixed MODIFY history_to VARCHAR(226);\
INSERT INTO history_mixed SELECT * FROM history;"
### Terminal 1
```

```
su -
ls -alt /var/lib/mysql/isfdb/history*
exit
### Terminal 2
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "Q:SELECT history_from, history_to
FROM isfdb.history_varchar" -t -i 200 -c 2 -q
./PT_Stress.php -u root -p oracle -s "Q:SELECT history_from, history_to
FROM isfdb.history_char" -t -i 200 -c 2 -q
./PT_Stress.php -u root -p oracle -s "Q:SELECT history_from, history_to
FROM isfdb.history" -t -i 200 -c 2 -q
./PT_Stress.php -u root -p oracle -s "Q:SELECT history_from, history_to
FROM isfdb.history_mixed" -t -i 200 -c 2 -q
mysql -uroot -poracle -e"ALTER TABLE isfdb.authors MODIFY \
author_birthdate DATE DEFAULT NULL"
```

Practice 9-3 Indexes

```
mysql -uroot -poracle -e"SELECT ROUTINE_NAME, ROUTINE_DEFINITION \
FROM INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME \
LIKE 'create_city_memory_huge'\G"
mysql -uroot -poracle -e"USE many_tables;\
CALL create_city_memory_huge(30);SELECT COUNT(*) FROM
city_memory_huge;"
mysql -uroot -poracle -e"SHOW CREATE TABLE \
many_tables.city_memory_huge\G"
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s btree-equal.sql -t
./PT_Stress.php -u root -p oracle -s hash-equal.sql -t
./PT_Stress.php -u root -p oracle -s scan-equal.sql -t
./PT_Stress.php -u root -p oracle -s btree-nonequal.sql -t
./PT_Stress.php -u root -p oracle -s hash-nonequal.sql -t
./PT_Stress.php -u root -p oracle -s scan-nonequal.sql -t
mysql -uroot -poracle -e"USE many_tables;\
CALL create_city_huge(30);SELECT COUNT(*) FROM city_huge;"
mysql -uroot -poracle -e"SHOW CREATE TABLE many_tables.city_huge\G"
su -
ls -alt /var/lib/mysql/many_tables/city_huge*
exit
mysql -uroot -poracle -e"ALTER TABLE many_tables.city_huge \
DROP INDEX Name, ADD INDEX (Name(4))"
mysql -uroot -poracle -e"SHOW CREATE TABLE many_tables.city_huge\G"
su -
ls -alt /var/lib/mysql/many_tables/city_huge*
exit
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s prefix-equal.sql -t
./PT_Stress.php -u root -p oracle -s prefix-nonequal.sql -t
```

Practice 9-4 Partitioning

```
mysql -uroot -poracle -e"CREATE TABLE many_tables.city_no_partition \
(ID int(11) NOT NULL, Name char(35) NOT NULL DEFAULT '', \
CountryCode char(3) NOT NULL DEFAULT '', \
District char(20) NOT NULL DEFAULT '', \
Population int(11) NOT NULL DEFAULT '0', \
PRIMARY KEY (ID, Population)) ENGINE=MYISAM"
mysql -uroot -poracle -e"INSERT INTO many_tables.city_no_partition \
SELECT * FROM many_tables.city_huge"
mysql -uroot -poracle -e"CREATE TABLE many_tables.city_partition \
(ID int(11) NOT NULL, Name char(35) NOT NULL DEFAULT '', \
CountryCode char(3) NOT NULL DEFAULT '', \
District char(20) NOT NULL DEFAULT '', \
Population int(11) NOT NULL DEFAULT '0') ENGINE=MYISAM \
PARTITION BY RANGE(Population) \
(PARTITION one VALUES LESS THAN (100000), \
PARTITION two VALUES LESS THAN (1000000), \
PARTITION three VALUES LESS THAN MAXVALUE)"
mysql -uroot -poracle -e"INSERT INTO many_tables.city_partition \
SELECT * FROM many_tables.city_huge"
mysql -uroot -poracle -e"ALTER TABLE many_tables.city_partition \
ADD PRIMARY KEY (ID, Population)"
mysql -uroot -poracle -e"SELECT partition_name, table_rows FROM \
INFORMATION_SCHEMA.PARTITIONS WHERE table_name = 'city_partition'"
cd /stage/scripts
./PT_Stress.php -u root -p oracle -s "Q:SELECT Name, Population \
FROM many_tables.city_no_partition WHERE Population
> 1000000" -t -i 10 -c 3 -q
./PT_Stress.php -u root -p oracle -s "Q:SELECT Name, Population FROM \
many_tables.city_partition WHERE Population
> 1000000" -t -i 10 -c 3 -q
./PT_Stress.php -u root -p oracle -s "Q:SELECT Name, Population FROM
many_tables.city_no_partition" -t -i 10 -c 3 -q
./PT_Stress.php -u root -p oracle -s "Q:SELECT Name, Population FROM
many_tables.city_partition" -t -i 10 -c 3 -q
```

Lesson 10: MySQL Query Performance

Practice 10-1 `EXPLAIN` Outputs

```
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM City \
WHERE ID=3803\G"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM City \
WHERE ID=3800+3\G"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM City \
WHERE ID-3=3800\G"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT Country.Name \
FROM Country, City WHERE City.CountryCode=Country.Code\G"
mysql -uroot -poracle -e"USE world;ALTER TABLE City \
ADD INDEX (District)"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM City \
WHERE District='California'\G"
mysql -uroot -poracle -e"USE world;DROP INDEX District ON City"
mysql -uroot -poracle -e"USE world;ALTER TABLE Country \
ADD INDEX (IndepYear)"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM Country \
WHERE IndepYear=1905 OR IndepYear IS NULL\G"
mysql -uroot -poracle -e"USE world;DROP INDEX IndepYear ON Country"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM City\G"
mysql -uroot -poracle -e"USE world;ALTER TABLE City \
ADD INDEX (District)"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM City \
WHERE ID=50 OR District='Michigan'\G"
mysql -uroot -poracle -e"USE world;DROP INDEX District ON City"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM City WHERE \
CountryCode IN (SELECT code FROM Country WHERE CODE LIKE 'USA')\G"
mysql -uroot -poracle -e"USE world;\
ALTER TABLE City ADD INDEX (Population)"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM City \
WHERE population>1000000\G"
mysql -uroot -poracle -e"USE world;DROP INDEX Population ON City"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT ID FROM City\G"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT * FROM City \
WHERE population>1000000\G"
```

Practice 10-2 Improve Query Executions

```
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT Name FROM City \
WHERE CountryCode = 'usa' ORDER BY Population DESC LIMIT 5\G"
mysql -uroot -poracle -e"USE world;ALTER TABLE City \
ADD INDEX (CountryCode)"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT Name FROM City \
WHERE CountryCode = 'usa' ORDER BY Population DESC LIMIT 5\G"
mysql -uroot -poracle -e"USE world;ALTER TABLE City \
ADD INDEX Code_Pop (CountryCode, Population)"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT Name FROM City WHERE
\
CountryCode = 'usa' ORDER BY Population DESC LIMIT 5\G"
```

```
mysql -uroot -poracle -e"USE world;ALTER TABLE City \
ADD INDEX Code_Pop_Name (CountryCode, Population, Name)"
mysql -uroot -poracle -e"USE world;EXPLAIN SELECT Name FROM City WHERE
\
CountryCode = 'usa' ORDER BY Population DESC LIMIT 5\G"
mysql -uroot -poracle -e"USE world;DROP INDEX CountryCode ON City"
mysql -uroot -poracle -e"USE world;DROP INDEX Code_Pop ON City"
mysql -uroot -poracle -e"USE world;DROP INDEX Code_Pop_Name ON City"
```

Practice 10-3 Locate Problematic Queries

```
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'log%'"
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'long%'"
mysql -uroot -poracle -e"SHOW VARIABLES LIKE 'slow%'"
mysql -uroot -poracle -e"SET GLOBAL slow_query_log_file = \
'/tmp/mysql_slow.log'"
mysql -uroot -poracle -e"SET GLOBAL long_query_time = 0.05"
mysql -uroot -poracle -e"SET GLOBAL slow_query_log = ON"
### Terminal 2
tail -f /tmp/mysql_slow.log
### Terminal 1
mysqlslap --user=root --password=oracle --delimiter=";" \
--query="/stage/scripts/isfdb_selects.sql" --create-schema="isfdb" \
--iterations=1
mysql -uroot -poracle
USE isfdb
SELECT DISTINCT
 (SELECT author_canonical FROM authors WHERE author_id=4)
 AS author, pub_title FROM pubs WHERE pub_id IN
 (SELECT pub_id FROM pub_authors WHERE author_id=4)
 ORDER BY pub_title;
EXPLAIN SELECT DISTINCT
 (SELECT author_canonical FROM authors WHERE author_id=4)
 AS author, pub_title FROM pubs WHERE pub_id IN
 (SELECT pub_id FROM pub_authors WHERE author_id=4)
 ORDER BY pub_title\G
SELECT DISTINCT author_canonical,pub_title
 FROM authors,pub_authors,pubs WHERE
 pub_authors.author_id=authors.author_id AND
 pub_authors.pub_id=pubs.pub_id AND authors.author_id=4
 ORDER BY pub_title;
SELECT "A's" AS LastName, COUNT(*)
 FROM authors WHERE author_lastname LIKE 'A%';
EXPLAIN SELECT "A's" AS LastName, COUNT(*)
 FROM authors WHERE author_lastname LIKE 'A%'\G
ALTER TABLE authors ADD INDEX (author_lastname(10));
SELECT "A's" AS LastName, COUNT(*)
 FROM authors WHERE author_lastname LIKE 'A%';
```

```
SELECT author_canonical,
 COUNT(award_id) AS Number_Awards FROM authors,
 title_awards, canonical_author
 WHERE canonical_author.author_id = authors.author_id
 AND title_awards.title_id = canonical_author.title_id
 GROUP BY author_canonical ORDER BY author_lastname;
EXPLAIN SELECT author_canonical,
 COUNT(award_id) AS Number_Awards FROM authors,
 title_awards, canonical_author
 WHERE canonical_author.author_id = authors.author_id
 AND title_awards.title_id = canonical_author.title_id
 GROUP BY author_canonical ORDER BY author_lastname\G
CREATE TEMPORARY TABLE award_count
 (author_id INT PRIMARY KEY, award_count INT)
 SELECT author_id, COUNT(award_id) AS award_count
 FROM canonical_author,title_awards
 WHERE title_awards.title_id = canonical_author.title_id
 GROUP BY author_id;
SELECT author_canonical, award_count FROM authors,
 award_count WHERE authors.author_id =
 award_count.author_id ORDER BY author_lastname;
```

Lesson 11: Performance Tuning Extras

No activities for this lesson.

Lesson 12: Conclusion

No activities for this lesson.

Basic Linux vi Editor Commands

What Is vi?

The default editor that comes with the UNIX operating system is called vi (visual editor). [Alternate editors for UNIX environments include pico and emacs, a product of GNU.]

The UNIX vi editor is a full screen editor and has two modes of operation:

- Command mode commands, which cause action to be taken on the file
- Insert mode, in which entered text is inserted into the file

In the command mode, every character typed is a command that does something to the text file being edited; a character typed in the command mode may even cause the vi editor to enter the insert mode. In the insert mode, every character typed is added to the text in the file; pressing the <Esc> (Escape) key turns off the Insert mode.

While there are a number of vi commands, just a handful of these are usually sufficient for beginning vi users. To assist such users, this appendix contains a sampling of basic vi commands. The most basic and useful commands are marked with an asterisk (* or star) in the following tables. With practice, these commands should become automatic.

Note: Both UNIX and vi are case-sensitive. Be sure not to use a capital letter in place of a lowercase letter; the results will not be what you expect.

To Get Into and Out of VI

To Start vi

To use vi on a file, enter `vi filename`. If the file named `filename` exists, then the first page (or screen) of the file will be displayed. If the file does not exist, then an empty file and screen are created into which you may enter text.

vi filename	*edit `filename` starting at line 1*
vi -r filename	*recover `filename` that was being edited when system crashed*

To Exit vi

Usually the new or modified file is saved when you leave vi. However, it is also possible to quit vi without saving the file.

Note: The cursor moves to the bottom of the screen whenever a colon (`:`) is typed. This type of command is completed by pressing the <Return> or <Enter> keys.

:x<Return>	*quit `vi`, writing out modified file to file named in original invocation*
:wq<Return>	*quit `vi`, writing out modified file to file named in original invocation*
:q<Return>	*quit (or exit) `vi`*
:q!<Return>	*quit `vi` even though latest changes have not been saved for this `vi` call*

Moving the Cursor

Unlike many of the PC and MacIntosh editors, the mouse does not move the cursor within the vi editor screen (or window). You must use the key commands listed below. On some UNIX platforms, the arrow keys may be used as well; however, since vi was designed with the Qwerty keyboard (containing no arrow keys) in mind, the arrow keys sometimes produce strange effects in vi and should be avoided.

If you go back and forth between a PC environment and a UNIX environment, you may find that this dissimilarity in methods for cursor movement is the most frustrating difference between the two.

In the table below, the symbol ^ before a letter means that the <Ctrl> key should be held down while the letter key is pressed.

j *or* <Return> [*or* down-arrow]	*move cursor down one line*
k [*or* up-arrow]	*move cursor up one line*
h *or* <Backspace> [*or* left-arrow]	*move cursor left one character*
l *or* <Space> [*or* right-arrow]	*move cursor right one character*
0 (zero)	*move cursor to start of current line (the one with the cursor)*
$	*move cursor to end of current line*
w	*move cursor to beginning of next word*
b	*move cursor back to beginning of preceding word*
:0<Return> *or* 1G	*move cursor to first line in file*
:n<Return> *or* nG	*move cursor to line n*
:$<Return> *or* G	*move cursor to last line in file*

Screen Manipulation

The following commands allow the vi editor screen (or window) to move up or down several lines and to be refreshed.

`^f`	*move forward one screen*
`^b`	*move backward one screen*
`^d`	*move down (forward) one half screen*
`^u`	*move up (back) one half screen*
`^l`	*redraws the screen*
`^r`	*redraws the screen, removing deleted lines*

Adding, Changing, and Deleting Text

Unlike PC editors, you cannot replace or delete text by highlighting it with the mouse. Instead, you use the commands in the following tables.

Perhaps the most important command is the one that allows you to back up and undo your last action. Unfortunately, this command acts like a toggle, undoing and redoing your most recent action. You cannot go back more than one step.

u	*UNDO WHATEVER YOU JUST DID; a simple toggle*

The main purpose of an editor is to create, add, or modify text for a file.

Inserting or Adding Text

The following commands allow you to insert and add text. Each of these commands puts the vi editor into insert mode; thus, the <Esc> key must be pressed to terminate the entry of text and to put the vi editor back into command mode.

i	*insert text before cursor, until <Esc> hit*
I	*insert text at beginning of current line, until <Esc> hit*
a	*append text after cursor, until <Esc> hit*
A	*append text to end of current line, until <Esc> hit*
o	*open and put text in a new line below current line, until <Esc> hit*
O	*open and put text in a new line above current line, until <Esc> hit*

Changing Text

The following commands allow you to modify text.

r	*replace single character under cursor (no <Esc> needed)*
R	*replace characters, starting with current cursor position, until <Esc> hit*
cw	*change the current word with new text, starting with the character under cursor, until <Esc> hit*
cNw	*change N words beginning with character under cursor, until <Esc> hit; e.g., c5w changes 5 words*
C	*change (replace) the characters in the current line, until <Esc> hit*
cc	*change (replace) the entire current line, stopping when <Esc> is hit*
Ncc ***or*** **cNc**	*change (replace) the next N lines, starting with the current line, stopping when <Esc> is hit*

Deleting Text

The following commands allow you to delete text.

x	*delete single character under cursor*
Nx	*delete N characters, starting with character under cursor*
dw	*delete the single word beginning with character under cursor*
dNw	*delete* `N` *words beginning with character under cursor;* *e.g.,* `d5w` *deletes 5 words*
D	*delete the remainder of the line, starting with current cursor position*
dd	*delete entire current line*
Ndd ***or*** **dNd**	*delete* `N` *lines, beginning with the current line;* *e.g.,* `5dd` *deletes 5 lines*

Cutting and Pasting Text

The following commands allow you to copy and paste text.

yy	*copy (yank, cut) the current line into the buffer*
Nyy ***or*** **yNy**	*copy (yank, cut) the next N lines, including the current line, into the buffer*
p	*put (paste) the line(s) in the buffer into the text after the current line*

Other Commands

Searching Text

A common occurrence in text editing is to replace one word or phase by another. To locate instances of particular sets of characters (or strings), use the following commands.

/string	*search forward for occurrence of* `string` *in text*
?string	*search backward for occurrence of* `string` *in text*
n	*move to next occurrence of search string*
N	*move to next occurrence of search string in opposite direction*

Determining Line Numbers

Being able to determine the line number of the current line or the total number of lines in the file being edited is sometimes useful.

:.=	*returns line number of current line at bottom of screen*
:=	*returns the total number of lines at bottom of screen*
^g	*provides the current line number, along with the total number of lines, in the file at the bottom of the screen*

Saving and Reading Files

These commands permit you to input and output files other than the named file with which you are currently working.

:r filename<Return>	*read file named* `filename` *and insert after current line (the line with cursor)*
:w<Return>	*write current contents to file named in original* `vi` *call*
:w newfile<Return>	*write current contents to a new file named* `newfile`
:12,35w smallfile<Return>	*write the contents of the lines numbered 12 through 35 to a new file named* `smallfile`
:w! prevfile<Return>	*write current contents over a pre-existing file named* `prevfile`