Test Note by Bhaskar Ranaware

Confidential Information

- *This Document is confidential to Infosys Technologies Limited. This document contains information and data that Infosys considers confidential and proprietary ("Confidential Information").*
- *Confidential Information includes, but is not limited to, the following:*
  - *Corporate and Infrastructure information about Infosys;*
  - *Infosys' project management and quality processes;*
  - *Project experiences provided included as illustrative case studies.*
- *Any disclosure of Confidential Information to, or use of it by a third party, will be damaging to Infosys.*
- *Ownership of all Infosys Confidential Information, no matter in what media it resides, remains with Infosys.*
- *Confidential information in this document shall not be disclosed, duplicated or used – in whole or in part – for any purpose without specific written permission of an authorized representative of Infosys.*

Course Objectives

- Database concepts – Recap
- Introduction to MySQL – Set up and using the same.
- DDL, DML and query statements – How to work on it.
- Advance queries
- Debugging

Session Plan

Day 1:

- Data Base Management System
- MYSQL
- MySQL Product Suite
- MySQL in web development
- MySQL Architecture – Broad View
- MySQL Server and Client
- Storage Engines, MySQL Architecture with Storage Engines
- SQL Basics
- SQL Statements: DDL

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

The plan may change.

Session Plan

Day 2:

- SQL Statements: DML, Transaction Control statement.
- Select Statement, Joins, SQL Functions
- Advanced SQL -- Subqueries, Views
- Procedures & Functions, Cursors
- Triggers
- Metadata information
- Debugging

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

The plan may change.

References

- "MySQL v 5.1 manual" – available at www.mysql.com.
- "MySQL Cookbook", Dubois Paul, SPD/ O'Reilly.
- "MySQL 5.0 Certification Study Guide", Dubois Paul, Hinz Stefan, Pedersen Carsten, MySQL Press
- "MySql Tutorial -- A Concise Introduction to the Fundamentals of working with MySql", Welling Luke, Thomson Laura, Pearson Education.

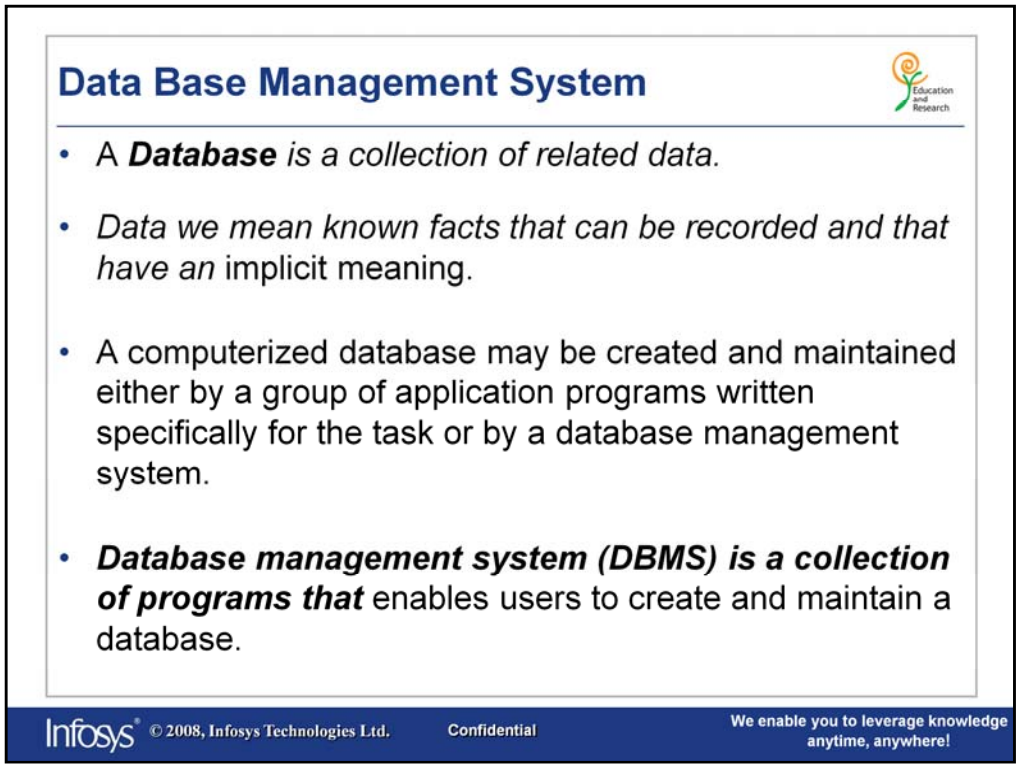
Infosys® © 2008, Infosys Technologies Ltd. Confidential

We enable you to leverage knowledge anytime, anywhere!

Education and Research

Part 1 – Introduction to MySQL

Infosys® © 2008, Infosys Technologies Ltd. Confidential



A ***database management system (DBMS) is a collection of programs that enables users to create and maintain a***

database.

The DBMS is hence a system that facilitates the processes of *defining, constructing and manipulating databases*

for various applications.

• Defining a database involves specifying the data types, structures and constraints for the data to be stored in

the database.

• Constructing the database is the process of storing the data itself on some storage media that is controlled

by the DBMS.

• Manipulating a database includes functions that query the database to retrieve specific data, updating data

to reflect changes in and around the application and generating reports from the data.

An important advantage of a DBMS is **data independence. In the traditional file system , data is dependent on**

application programs. Any changes on the data are made through application programs. Where as DBMS

allows the application developer to change the data structure without affecting the application as both are

built independent of one another.

Relational DBMS

- A system that is able to manage its database entirely through its relational capabilities is called a **Relational Database Management System (RDBMS).**
- Data files are structured as relations also called as tables.
- Tables are a collection of rows (or tuples).
- A relational database is a database divided into logical units called tables, where tables are related to one another within the database.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Relational Database Management Systems

A system that is able to manage its database entirely through its relational capabilities is called a **Relational**

Database Management System (RDBMS). For a DBMS to be classified as a RDBMS it should satisfy

Codd's twelve rules proposed for the relational model.

Operations of Relational Algebra

Commonly used Terms in RDBMS

- **Table:** Tables are used storing data. A database can have many different tables.
- **Fields:** Also called as columns.
- **Records (tuples):** Is a particular instance of the data object.
- **Query:** Queries are used to retrieve data from database. Queries allow data to be combined from different tables, find, sort, do calculations. Can also be used to create *custom views of the data.*

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Advantages of RDBMS

- Controlling Redundancy
- Restricting Unauthorized Access
- Enforcing Integrity Constraints
- Representing Complex relationships among Data
- Provide Backup and Recovery
- Flexibility
- Data Independence

Infosys® © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Application of RDBMS

- Traditional Database Applications:
  Ex: Employee Management System
  Payroll System

- Web Applications
  Ex: Email application
  Online banking application

- Web Services
  Ex: Amazon / Flipkart

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

MYSQL

- **MySQL** is a relational database management system.
- **MySQL**, the most popular Open Source SQL DBMS is developed, distributed, and supported by Sun Microsystems, Inc.
- Licensed with the **GNU** General public license http://www.gnu.org/
- MySQL Database Server is very fast, reliable, and easy to use.
- The MySQL Database Software is a client/server software.
- Supported on almost all platforms including Solaris, RedHat Linux, Windows, OpenBSD etc.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

HISTORY

- MySQL was originally developed by Michael Widenius and David Axmark.
- Started with mSQL (Mini SQL) – a light-weight SQL Database engine technology, developed in 1994.
- But later found out that mSQL was not fast and flexible enough to face their needs.
- The result is a new SQL server - **MySQL**
- MySQL is named after co-founder Monty Widenius's daughter, My.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

MySQL Product Suite

- MySQL Enterprise : The MySQL Enterprise offers production support, monitoring tools and MySQL database software to deliver optimal performance, reliability, security, and uptime. It includes:
  - MySQL Enterprise Server
  - MySQL Enterprise Monitor
  - MySQL Production Support
- MySQL Cluster
- MySQL Embedded Database
- MySQL Connectors
- MySQL Workbench

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Reference: http://www.mysql.com/products

MySQL Enterprise Server software is the most reliable, secure and up-to-date version of MySQL for cost-effectively delivering E-commerce, Online Transaction Processing (OLTP), and multi-terabyte Data Warehousing applications. It is a fully integrated transaction-safe, ACID compliant database with full commit, rollback, crash recovery and row level locking capabilities. MySQL delivers the ease of use, scalability, and performance that has made MySQL the world's most popular open source database.

The MySQL Enterprise Monitor continuously monitors your MySQL servers and alerts you to potential problems before they impact your system. Its like having a "Virtual DBA Assistant" at your side to recommend best practices to eliminate security vulnerabilities, improve replication, optimize performance and more. As a result, the productivity of your developers, DBAs and System Administrators is improved significantly.

MySQL Enterprise includes 24x7 Production Support that helps you deliver continuous availability for your production database applications. MySQL Production Support gives you priority access with fast response times to assist you with the development, deployment and management of your MySQL applications. MySQL Enterprise gives you the flexibility to choose a service level (Basic, Silver, Gold, Platinum) that matches your requirements.

MySQL Cluster

MySQL Cluster delivers a fault tolerant database clustering architecture for deploying highly available mission-critical database applications.

MySQL Embedded Database

The MySQL Embedded Database is the most popular choice for OEMs/ISVs who want to cost-effectively embed or bundle a reliable and high-performance relational database.

MySQL Connectors

MySQL provides standards-based database drivers for JDBC, ODBC, and .Net enabling developers to build database applications in their language of choice. In addition, a native C library allows developers to embed MySQL directly into their applications.

MySQL Workbench

MySQL Workbench enables a DBA, developer, or data architect to visually design, generate, and manage all types of databases including Web, OLTP, and data warehouse databases. It includes everything a data modeler needs for creating complex ER models, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort.

MySQL Internals:

- Written in C and C++. Uses GNU Automake, Autoconf, and Libtool for portability.
- Designed to be fully multi-threaded using kernel threads, to easily use multiple CPUs if they are available.
- Provides transactional and non-transactional storage engines. Designed to make it relatively easy to add other storage engines.
- Uses very fast B-tree disk tables (MyISAM) with index compression. Executes very fast joins using an optimized one-sweep multi-join.
- Implements in-memory hash tables, which are used as temporary tables.
- Meets the ANSI SQL92 regulations (SQL-Structured Query Language).

Infosys® © 2008, Infosys Technologies Ltd. Confidential

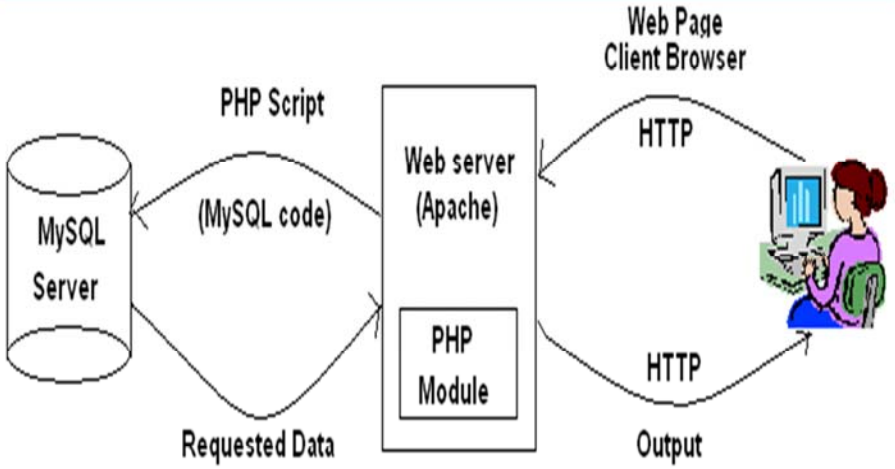
We enable you to leverage knowledge anytime, anywhere!

MySQL in web development:

- With the help of a web server, a MySQL database server and some scripting language like PHP we can develop a large dynamic web sites.
- The web application can range from complex distribution system to guest books, discussion boards etc.
- **MySQL** is a small, compact and easy to use database server.
- PHP with MySQL has become popular over the time for web development with databases.

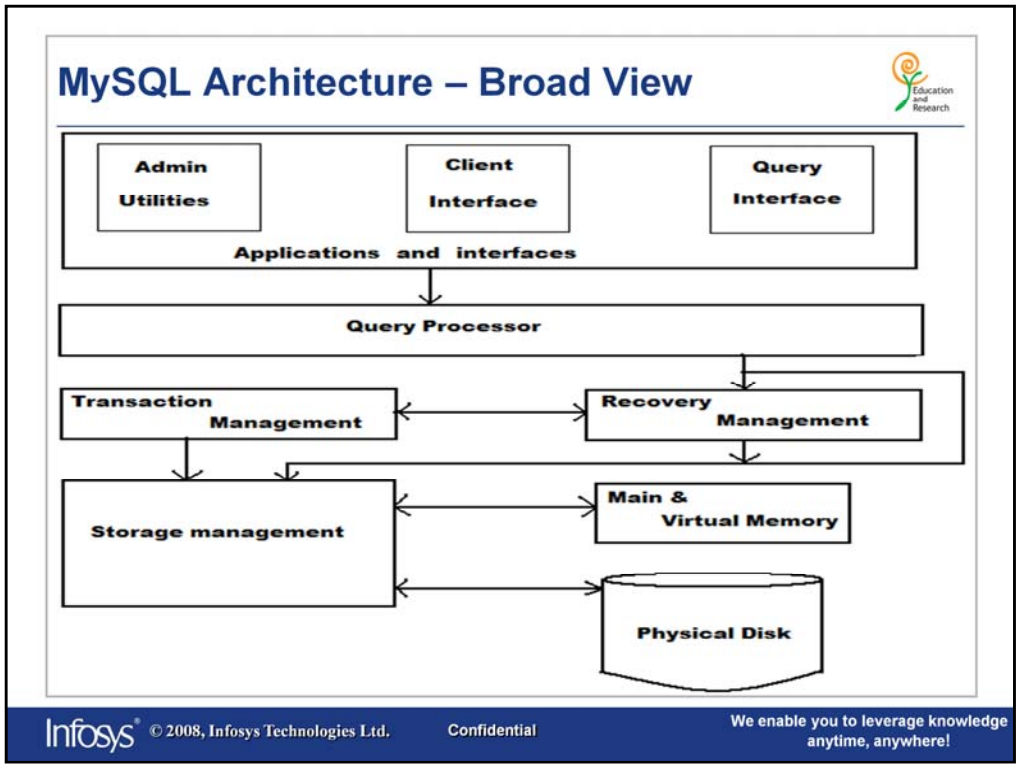
Infosys® © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

PHP and MySQL



Infosys © 2008, Infosys Technologies Ltd. Confidential

We enable you to leverage knowledge anytime, anywhere!



Ref: **MySQL Conceptual Architecture by** Ryan Bannon, Alvin Chin, Faryaaz Kassam and Andrew Roszko

MySQL Installation:

- Source code and binaries are available at http://www.mysql.com
- You can download the installable for Windows / Unix / Linux. The MySQL comes in version 5.0, 5.1 and 5.4.
- Download the source or binaries as required. The source code has to be compiled, hence better suited for Linux / Unix environment.
- Execute setup.exe or .msi file
- Destination Folder: Default
  (C:\Program Files\MySQL\MySQL Server 5.1\)

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Version 5.4 was still under beta release during the development of these slides. So for this course we will use v5.1.

MySQL Installation:

To run MySQL on Windows, you need the following:

- Windows 32-bit and 64-bit versions are supported.
- Should have permission to run MySQL server as a service.
- Install MySQL on Windows using an account that has administrator rights.
- TCP/IP protocol support.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Under Administrator, Otherwise, you may encounter problems with certain operations such as editing the PATH environment variable or accessing the **Service Control Manager**.

MySQL Server:

- MySQL server comes as follows:
  - mysqld
  - mysqld –debug
- To start the server, enter "**mysqld –console"** at command prompt or start the mysql service in case of windows.
- Following would indicate that the server is ready to accept client connections (in case of command prompt):

  mysqld: ready for connections Version: '5.1.40' socket: '' port: 3306

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

MySQL Server:

- To shutdown the MySQL server use:

  mysqladmin -u root shutdown

- To test whether MySQL server is working use any one of the following
  - Mysqlshow
  - mysqlshow" -u root mysql
  - mysqladmin version status
  - mysql test

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Environment Variables:

Following is the short list of environment variables used by MySQL:

MYSQL_HISTFILE	The path to the mysql history file.
MYSQL_HOME	The path to the directory in which the server-specific my.cnf file resides.
MYSQL_HOST	The default host name used by the mysql command-line client.
MYSQL_PS1	The command prompt to use in the mysql command-line client.
MYSQL_PWD	The default password when connecting to mysqld. Using this is insecure.
TMPDIR	The directory where temporary files are created.
TZ	This should be set to your local time zone.
USER	The default user name on Windows, used when connecting to mysqld.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

MySQL Client:

- To Connect to a MySQL server use:

  mysql -h *host -u user –p*

  where host: host name where MySQL server is running
  and user: user name of MySQL account

  After connection is successful the

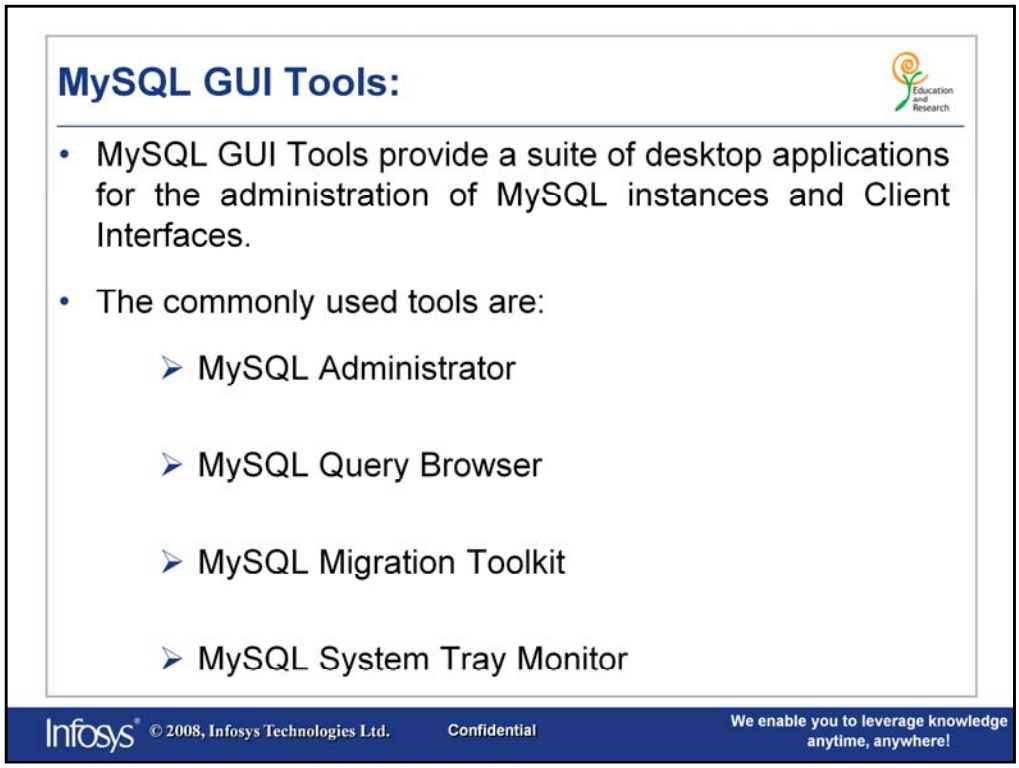
  mysql> prompt

  will tells you that mysql is ready to accept commands.

- To disconnect just say:

  quit

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!



Reference: MySQL reference manual

The MySQL GUI Tools provide a suite of desktop applications for the administration of MySQL instances and for building and manipulating the data within MySQL databases.

MySQL Administrator – provides administration tools and utilities to monitor and administer databases, users privileges and data within a MySQL instance.

MySQL Query Browser – provides an interface for building and testing queries interactively.

MySQL Migration Toolkit – used to migrate existing database information to MySQL (Windows only).

MySQL Programs:

Along with the server (mysqld) and client (mysql) programs following are the other programs in MySQL installation:

Server Programs:

- **mysqld_safe**: A server startup script. mysqld_safe attempts to start mysqld.
- **mysql.server**: A server startup script. It invokes mysqld_safe to start the MySQL server.
- **mysqld_multi**: A server startup script that can start or stop multiple servers installed on the system.
- **mysqlmanager**: It is the MySQL Instance Manager and an alternative to mysqld_multi.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

MySQL Programs:

Installation Programs:

- **mysql_install_db**: This script creates the MySQL database and initializes the grant tables with default privileges. It is usually executed only once, during first installation.
- **mysql_secure_installation:** This program enables you to improve the security of your MySQL installation.
- **mysql_upgrade:** This program is used after a MySQL upgrade operation. It checks tables for incompatibilities and repairs them if necessary, and updates the grant tables with any changes that have been made in newer versions of MySQL.

Infosys® © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

MySQL Programs:

Client Programs:

- **mysqladmin:** Performs administrative operations (such as creating/dropping databases, flushing tables to disk, reopening log files etc. It can also be used to retrieve version, process, and status information from the server.
- **mysqldump:** A client that dumps a MySQL database into a file as SQL, text, or XML.
- **mysqlimport:** A client that imports text files into their respective tables using LOAD DATA INFILE.
- **mysqlshow:** A client that displays information about databases, tables, columns, and indexes.

Infosys® © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Education and Research

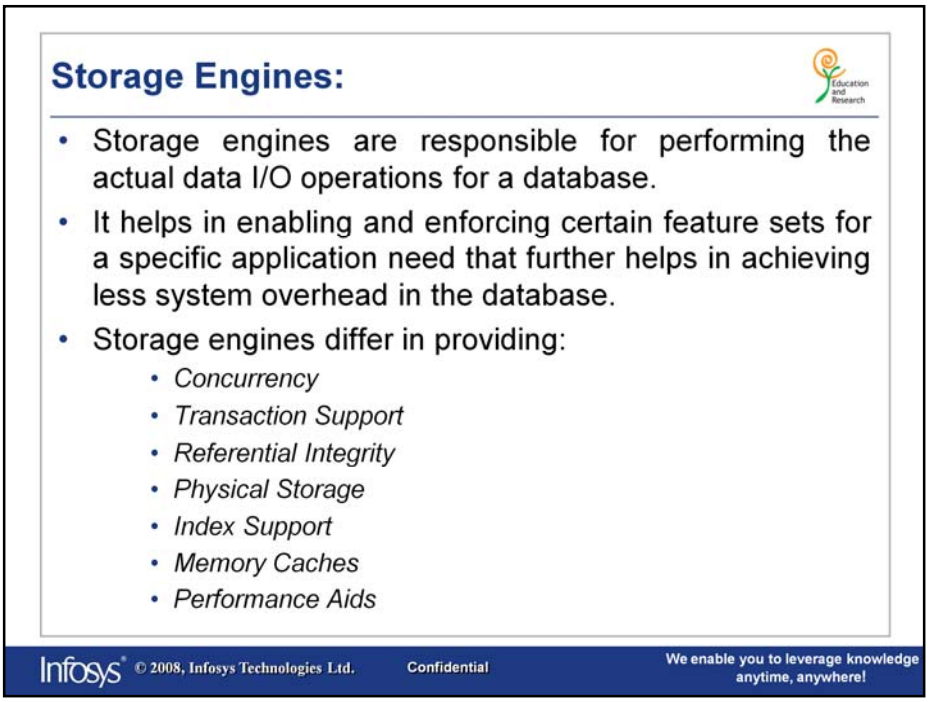
Part 2 – Storage Engines

Infosys® © 2008, Infosys Technologies Ltd. Confidential

Storage Engines

- A storage engine is a low level data storage / retrieval module (disk or memory).
- The MySQL pluggable storage engine architecture allows a database professional to select a specialized storage engine for a particular application need.
- The MySQL server architecture isolates the application programmer and DBA from all of the low-level implementation details at the storage level, providing a consistent and easy application model and API.
- MySQL supported multiple storage engines.

Infosys © 2008, Infosys Technologies Ltd. Confidential



Ref: MySQL reference guide – available at www.mysql.com

A MySQL pluggable storage engine is the component in the MySQL database server that is responsible for performing the actual

data I/O operations for a database as well as enabling and enforcing certain feature sets that target a specific application need. A

major benefit of using specific storage engines is that you are only delivered the features needed for a particular application, and

therefore you have less system overhead in the database, with the end result being more efficient and higher database performance.

This is one of the reasons that MySQL has always been known to have such high performance, matching or beating proprietary

monolithic databases in industry standard benchmarks.

From a technical perspective, what are some of the unique supporting infrastructure components that are in a storage engine? Some

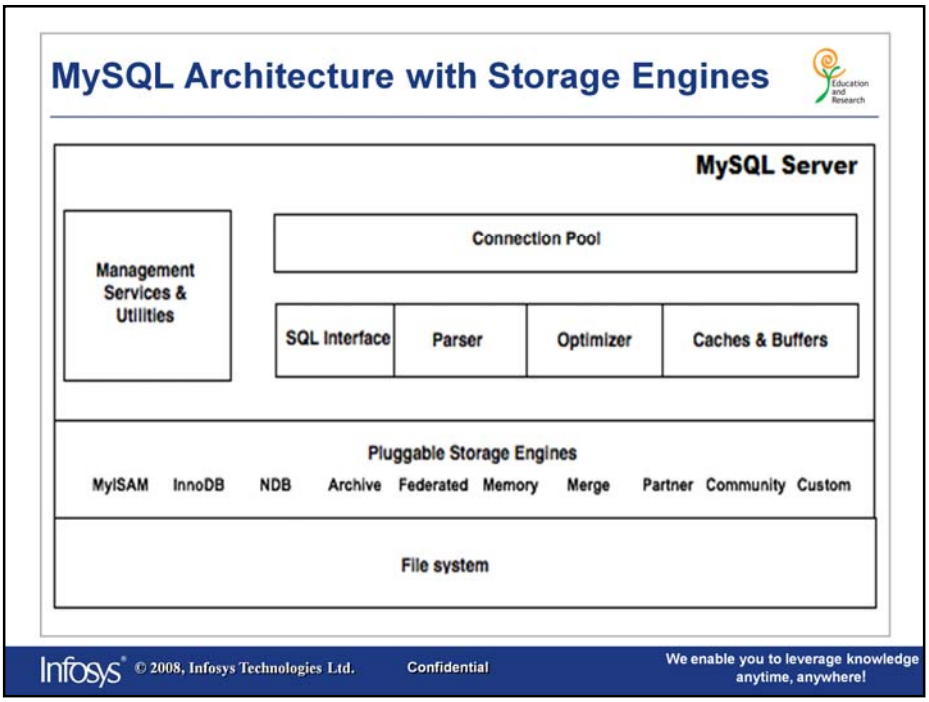
of the key feature differentiations include:

• *Concurrency — some applications have more granular lock requirements (such as row-level locks) than others. Choosing the*

right locking strategy can reduce overhead and therefore improve overall performance. This area also includes support for capabilities

such as multi-version concurrency control or “snapshot” read.

• *Transaction Support — Not every application needs transactions, but for those*



Ref: MySQL reference guide – available at www.mysql.com

Storage Engines Types:

MySQL 5.1 supports the following storage engines:

- MyISAM — The default MySQL storage engine and the one that is used the most in Web, data warehousing, and other application and is the default storage engine.
- InnoDB — A transaction-safe (ACID compliant) storage engine for MySQL that has commit, rollback, and crash-recovery capabilities to protect user data.
- Memory(formerly heap engine) — Stores all data in RAM for extremely fast access in environments that require quick lookups of reference and other data.
- Merge — Allows a MySQL DBA or developer to logically group a series of identical MyISAM tables and reference them as one object. Good for VLDB environments such as data warehousing.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Ref: MySQL reference guide – available at www.mysql.com

A MySQL pluggable storage engine is the component in the MySQL database server that is responsible for performing the actual

data I/O operations for a database as well as enabling and enforcing certain feature sets that target a specific application need. A

major benefit of using specific storage engines is that you are only delivered the features needed for a particular application, and

therefore you have less system overhead in the database, with the end result being more efficient and higher database performance.

This is one of the reasons that MySQL has always been known to have such high performance, matching or beating proprietary

monolithic databases in industry standard benchmarks.

From a technical perspective, what are some of the unique supporting infrastructure components that are in a storage engine? Some

of the key feature differentiations include:

• *Concurrency — some applications have more granular lock requirements (such as row-level locks) than others. Choosing the*

right locking strategy can reduce overhead and therefore improve overall performance. This area also includes support for capabilities

such as multi-version concurrency control or “snapshot” read.

• *Transaction Support — Not every application needs transactions, but for those*

Storage Engines Types:

- Archive — Provides the perfect solution for storing and retrieving large amounts of seldom-referenced historical, archived, or security audit information.
- Federated — Offers the ability to link separate MySQL servers to create one logical database from many physical servers. Very good for distributed or data mart environments.
- NDB — The Clustered database engine that is particularly suited for applications with high performance lookup needs that also require the highest possible degree of uptime and availability.
- CSV — The CSV storage engine stores data in text files using comma-separated values format.

Other includes:

Blackhole, Example etc.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Ref: MySQL reference guide – available at www.mysql.com

A MySQL pluggable storage engine is the component in the MySQL database server that is responsible for performing the actual

data I/O operations for a database as well as enabling and enforcing certain feature sets that target a specific application need. A

major benefit of using specific storage engines is that you are only delivered the features needed for a particular application, and

therefore you have less system overhead in the database, with the end result being more efficient and higher database performance.

This is one of the reasons that MySQL has always been known to have such high performance, matching or beating proprietary

monolithic databases in industry standard benchmarks.

From a technical perspective, what are some of the unique supporting infrastructure components that are in a storage engine? Some

of the key feature differentiations include:

• *Concurrency — some applications have more granular lock requirements (such as row-level locks) than others. Choosing the*

right locking strategy can reduce overhead and therefore improve overall performance. This area also includes support for capabilities

such as multi-version concurrency control or “snapshot” read.

• *Transaction Support — Not every application needs transactions, but for those*

Education and Research

Part 3 – SQL Basics

Infosys® © 2008, Infosys Technologies Ltd. Confidential

Database objects:

- Any defined object in a database that is used to store or reference data.
- Considered to be backbone of relational databases.
- Examples includes tables, views, clusters, sequences, indexes etc.
- Schema is a collection of database objects associated with one particular database username.
- In our example root is the user name and there is inbuilt schema called test.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Entering queries:

- Need to be connected. Use mysql command to connect to the server.
- Queries are normal SQL statement that ends with a ;. They are entered at the mysql prompt or in the query browser tool.
- MySQL will send the query to the server and return the result in the form of row and columns.
- Example query:
  - **select user(), version(), current_date;**
  - This will display the user name, version of the mysql and current date.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

SQL Statements:

Following are the basic types of SQL statements:

- Data definition language (DDL) statements
- Data manipulation language (DML) statements
- Transaction Control Statements

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL Statements:

Following are some of the DDL statements:

- ALTER
- ANALYZE
- CREATE
- DROP
- RENAME
- REVOKE
- TRUNCATE

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Create Database Statement

- Databases are used to store, manage and retrieve information. They use tables for this purpose.

- To create database with a given name, we use CREATE DATABASE syntax as below:

  CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] *database_name;*

  Example:

  Create database if not exists Employee_DataBase;

- Using [IF NOT EXISTS], suppresses an error if a database exists with the same name.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Create Database Statement

- Database characteristics are stored in the db.opt file in the database directory.
- A database in MySQL is implemented as a directory containing files that correspond to tables in the database.
- CREATE DATABASE statement creates only a directory under the MySQL data directory and the db.opt file when it is initially created.
- If you manually create a directory under the data directory, the server considers it a database directory and it shows up in the output of SHOW DATABASES.

Infosys® © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Queries on databases

- **SHOW DATABASES:** To find out what databases currently exist on the server. It does not show databases, that you have no privileges.
- **SELECT DATABASE():** Returns the default or current database name. If there is no default database, returns NULL.
- To select a database use:
  - **mysql *database_name;*** (This will connect to server as well as select the database for your use)
  - ***Use database_name;*** (This can be used after the connection have been established to use the required database)

Ex: **mysql employee_database;**
use employee_database;

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Create Tablespace Statement

- A tablespace is a storage location where the actual data underlying database objects can be kept. It is the physical portion of the database used to allocate storage for all segments such as tables, indexes and rollback.
- Tablespace specify only the database storage locations, not the logical database structure.
- Different objects in the same schema may have different underlying tablespaces. Similarly, a tablespace may service segments for more than one schema.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Create Tablespace Statement

Syntax:

```
CREATE TABLESPACE tablespace_name
    ADD DATAFILE 'file_name'
    USE LOGFILE GROUP logfile_group
    [EXTENT_SIZE [=] extent_size]
    [INITIAL_SIZE [=] initial_size]
    [AUTOEXTEND_SIZE [=] autoextend_size]
    [MAX_SIZE [=] max_size]
    [NODEGROUP [=] nodegroup_id]
    [WAIT] [COMMENT [=] comment_text]
    ENGINE [=] engine_name;
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Create Tablespace Statement

- **Datafile:** Tablespace contain one or more data files that provides storage space for tables. Using this option one data file is created and added to the tablespace.
- **Logfile Group:** It associates a log file group of one or more UNDO log files to the tablespace. logfile_group must be created first.
- **EXTENT_SIZE:** It sets the size (in bytes) of the extents used by any files belonging to the tablespace. The default value is 1M.
- **INITIAL_SIZE:** Sets the data file's total size in bytes. The size can't be changed once the file has been created but more data files can be added. The default value is 128M.
- **ENGINE:** Determines the storage engine to be used.
- All other parameters are parsed but have no effect in MySQL 5.1 and are intended for future expansion.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Ref: MySQL reference guide – available at www.mysql.com

CREATE LOGFILE GROUP *: Is used to create logfile group.*

In most cases, changing the extent size does not have any measurable effect on performance, and the default value is recommended

for all but the most unusual situations. An *extent is a unit of disk space allocation. One extent is filled with as much data as that extent can contain before another extent is* used. In theory, up to 65,535 (64K) extents may used per data file; however, the recommended maximum is 32,768 (32K). The recommended maximum size for a single data file is 32G — that is, 32K extents × 1 MB per extent. In addition, once an extent is allocated to a given partition, it cannot be used to store data from a different partition; an extent cannot store data from more than one partition. This means, for example that a tablespace having a single datafile whose INITIAL_SIZE is 256 MB and whose EXTENT_

SIZE is 128M has just two extents, and so can be used to store data from at most two different disk data table partitions.

DDL – Create Table Statement

- A table or a relation is a single store of related information. A table consists of records, and each record is made up of a fields.
- CREATE TABLE creates a table with the given name.
- By default, the table is created in the default database. An error occurs if the table exists, if there is no default database, or if the database does not exist.
- The table name can be specified as *db_name.tbl_name to create the table in a specific database.*

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Create Table Statement

Syntax:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table*name* (*create_parameter)* [*table_parameter]* [*partition_parameter]*

Or:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table*name* [(*create_parameter)]* [*table_parameter]* [*partition_parameter]* *select_statement*

Or:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table*name* { LIKE *old_table_name* | *(LIKE old_table_name) }*

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Create Table Statement

- To create the table in a specific database we can specify the table name as : *database_name.table_name.*
- The TEMPORARY keyword is used to create a temporary table that is visible only to the current connection, and is dropped automatically when the connection is closed.
- The keywords IF NOT EXISTS prevent an error from occurring if the table with the same name exists.
- MySQL represents each table by an .frm table format (definition) file in the database directory. The storage engine for the table might create other files as well.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

In the case of MyISAM tables, the storage engine creates data and index files. Thus, for each MyISAM table *table_name, there are three disk files.*

Create Table – Create Parameters

Create Parameters are :

- **Column_name:** Any valid identifier can work as a column name. All column names must be unique within the table.
- **Column_definition:** Contains the information of the data types.
- **Constraint:** Constraints are used to limit the type of data that can go into a table. It is specified when the table is first created.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Data Types:

Data types can be classified into following categories:

- **Numeric:** bit, tinybit, tinyint, smallint, mediumint, int, integer, bigint, real, double, float, decimal, numeric, boolean, bool
- **Strings:** char, varchar, text, mediumtext, longtext, binary, varbinary, tinyblob, blob, mediumblob, longblob, enum, set
- **Date/Time:** date, time, timestamp, datetime, year

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Data Types – Numeric Type

Data Type	Size (In Bytes)
TINYINT	1
SMALLINT	2
MEDIUMINT	3
INT, INTEGER	4
BIGINT	8
FLOAT(*p*)	4 bytes if 0 <= *p* <= 24, 8 bytes if 25 <= *p* <= 53
FLOAT	4
DOUBLE [PRECISION], REAL	8
DECIMAL(*M*,*D*), NUMERIC(*M*,*D*)	Varies
BIT(*M*)	approximately (M+7)/8 bytes

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Data Types – Numeric Type

```
CREATE TABLE table1 (
        id INT,
        salary FLOAT,
        manager BIT(1)
    );
CREATE TEMPORARY TABLE table1 (
        id INT,
        salary FLOAT,
        manager BIT(1)
    );
CREATE TEMPORARY TABLE IF NOT EXISTS table1 (
        id INT,
        salary FLOAT,
        manager BIT(1)
    );
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Data Types – Date and Time

- **DATETIME**: Used to store values that contain both date and time information.
- **TIMESTAMP:** Used when we need automatic initialization and updating of date and time field.
- **YEAR: Used for** representing years. Can be used as YEAR(2) or YEAR(4) to specify a display width with default been four characters.

Data Type	Size (In Bytes)
DATE	3
TIME	3
DATETIME	8
TIMESTAMP	4
YEAR	1

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Data Types – Strings

- **CHAR and VARCHAR**: Both are used to strings. The length of a CHAR column is fixed to the length given at the time of table creation and can be vary from 0 – 255. In VARCHAR columns are variable-length strings. The length can vary from 0 to 65,535.

  Example: To store 'deepak' in CHAR(8) declaration would use 8 bytes, where as in VARCHR(8) it would use only 6 bytes.

- **BINARY and VARBINARY:** They are similar to CHAR and VARCHAR and contains contain byte strings rather than character strings.

- **BLOB and TEXT:** BLOB is a binary large object that can hold a variable amount of data. The corresponding character object is TEXT. BLOB columns contains byte strings where as TEXT columns contains character strings.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Ref: MySQL reference available at www.mysql.com

VARCHAR: The effective maximum length of a VARCHAR is subject to the maximum row size (65,535 bytes, which is shared among all columns) and the character set used.

In contrast to CHAR, VARCHAR values are stored as a one-byte or two-byte length prefix plus data. The length prefix indicates the

number of bytes in the value. A column uses one length byte if values require no more than 255 bytes, two length bytes if values

may require more than 255 bytes.

If strict SQL mode is not enabled and you assign a value to a CHAR or VARCHAR column that exceeds the column's maximum

length, the value is truncated to fit and a warning is generated. For truncation of nonspace characters, you can cause an error to occur

(rather than a warning) and suppress insertion of the value by using strict SQL mode. For VARCHAR columns, trailing spaces in excess of the column length are truncated prior to insertion and a warning is generated, regardless of the SQL mode in use. For CHAR columns, truncation of excess trailing spaces from inserted values is performed silently regardless of the SQL mode.

VARCHAR values are not padded when they are stored. Trailing spaces are retained when values are stored and retrieved, in conformance

with standard SQL.

BLOB columns have no character set, and sorting and comparison are based on the

Data Types – Strings

- **ENUM:** The column values are chosen from a list of allowed values that are enumerated explicitly mentioned at the table creation time. The enum value has to be a quoted string literal and can't be an expression. For Example:

```
CREATE TABLE table1 (
        id INT,    salary FLOAT, manager BIT(1),
        role enum ('SE', 'SSE', 'TA')     );
```

  The values in enum are given an index as per there position. There could be maximum of 65535 elements in an enum.

- **SET:** A column of type set can have zero or more values, each of which chosen from a list of allowed values specified when the table is created. For example:

```
CREATE TABLE table1 (grades  SET('a+', 'a', 'b', 'c'));
```

  In this case, you can store the values like 'a+,c', 'b,a', etc.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Ref: MySQL reference available at www.mysql.com

VARCHAR: The effective maximum length of a VARCHAR is subject to the maximum row size (65,535 bytes, which is shared among all columns) and the character set used.

In contrast to CHAR, VARCHAR values are stored as a one-byte or two-byte length prefix plus data. The length prefix indicates the

number of bytes in the value. A column uses one length byte if values require no more than 255 bytes, two length bytes if values

may require more than 255 bytes.

If strict SQL mode is not enabled and you assign a value to a CHAR or VARCHAR column that exceeds the column's maximum

length, the value is truncated to fit and a warning is generated. For truncation of nonspace characters, you can cause an error to occur

(rather than a warning) and suppress insertion of the value by using strict SQL mode. For VARCHAR columns, trailing spaces in excess of the column length are truncated prior to insertion and a warning is generated, regardless of the SQL mode in use. For CHAR columns, truncation of excess trailing spaces from inserted values is performed silently regardless of the SQL mode.

VARCHAR values are not padded when they are stored. Trailing spaces are retained when values are stored and retrieved, in conformance

with standard SQL.

BLOB columns have no character set, and sorting and comparison are based on the

Create Table -- Examples

- CREATE TABLE tableEx1 (id INT, name VARCHAR(10)) TYPE=innodb;
- CREATE TABLE tableEx2 (id INT, name VARCHAR(10), current_timestamp TIMESTAMP(8));
- CREATE TABLE car (id INT, name VARCHAR(10), color ENUM('red', 'blue', 'green', 'yellow'), options SET('rack', 'light', 'helmet', 'lock'));
- CREATE TABLE book (id INT, name VARCHAR(10), copyright YEAR, date_written TIMESTAMP));

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Create Table – Table Parameters

Following makes the table options part of syntax:

ENGINE [=] *engine_name*
AUTO_INCREMENT [=] *value*
AVG_ROW_LENGTH [=] *value*
COMMENT [=] '*string*'
DATA DIRECTORY [=] '*absolute path to directory*'
INSERT_METHOD [=] { NO | FIRST | LAST }
MAX_ROWS [=] *value*
MIN_ROWS [=] *value*
PASSWORD [=] '*string*'
TABLESPACE *tablespace_name*
UNION [=] (*tbl_name[,tbl_name]*

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Ref: MySQL Reference Manual available at www.mysql.com

AUTO_INCREMENT: The initial AUTO_INCREMENT value for the table. In MySQL 5.1, this works for MyISAM, MEMORY, and InnoDB tables.

AVG_ROW_LENGTH: An approximation of the average row length for your table. You need to set this only for large tables with variable-size rows.

COMMENT: A comment for the table, up to 60 characters long.

DATA DIRECTORY, INDEX DIRECTORY: By using DATA DIRECTORY='*directory' or INDEX DIRECTORY='directory' you can specify where the MyISAM* storage engine should put a table's data file and index file. The directory must be the full path name to the directory, not a

relative path.

INSERT_METHOD: If you want to insert data into a MERGE table, you must specify with INSERT_METHOD the table into which the row should be inserted. INSERT_METHOD is an option useful for MERGE tables only. Use a value of FIRST or LAST to have inserts go to

the first or last table, or a value of NO to prevent inserts.

MAX_ROWS: The maximum number of rows you plan to store in the table. This is not a hard limit, but rather a hint to the storage engine that the table must be able to store at least this many rows.

MIN_ROWS: The minimum number of rows you plan to store in the table. The MEMORY storage engine uses this option as a hint about

memory use.

PASSWORD: This option is unused

Create Table – Partition Parameters

Following makes the partition options part of syntax:

```
PARTITION BY { [LINEAR] HASH(expr) | [LINEAR]
   KEY(column_list)  | RANGE(expr)  | LIST(expr) }
[PARTITIONS num]
PARTITION partition_name
[VALUES {LESS THAN {(expr) | MAXVALUE} | IN (value_list)}]
[[STORAGE] ENGINE [=] engine_name]
[COMMENT [=] 'comment_text' ]
[DATA DIRECTORY [=] 'data_dir']
[INDEX DIRECTORY [=] 'index_dir']
[MAX_ROWS [=] max_number_of_rows]
[MIN_ROWS [=] min_number_of_rows]
[TABLESPACE [=] tablespace_name]
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

HASH(*expr): Hashes one or more columns to create a key for placing and locating rows. expr is an expression using one or*

more table columns. This can be any legal MySQL expression (including MySQL functions) that yields a single integer value.

The LINEAR keyword entails a somewhat different algorithm. In this case, the number of the partition in which a row is stored

is calculated as the result of one or more logical AND operations.

KEY(column_list): This is similar to HASH, except that MySQL supplies the hashing function so as to guarantee an even

data distribution. The column_list argument is simply a list of table columns.

RANGE: In this case, *expr shows a range of values using a set of VALUES LESS THAN operators. When using range partitioning,*

you must define at least one partition using VALUES LESS THAN. You cannot use VALUES IN with range partitioning.

VALUES LESS THAN can be used with either a literal value or an expression that evaluates to a single value.

LIST(*expr): This is useful when assigning partitions based on a table column with a restricted set of possible values, such as*

a state or country code. In such a case, all rows pertaining to a certain state or

Create Table – Partition Examples

- CREATE TABLE tableEx1 (id INT) PARTITION BY HASH(id);
- CREATE TABLE tableEx2 (id INT, dtob DATETIME) PARTITION BY HASH (YEAR(dtob));
- CREATE TABLE tableEx3 (id INT, doj DATE) PARTITION BY LINEAR KEY(doj) PARTITIONS 5;
- CREATE TABLE tableEx4 (id INT, doj DATE)
  PARTITION BY RANGE (doj) (
  PARTITION p1 VALUES LESS THAN (1991),
  PARTITION p2 VALUES LESS THAN (2001),
  PARTITION p3 VALUES LESS THAN MAXVALUE);

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Create Table – Constraints

- Constraints are used to limit the type of data that can go into a table.
- Common types of constraints include the following:
  - NOT NULL: Ensures that a column cannot have NULL value.
  - DEFAULT: Provides a default value for a column when none is specified.
  - UNIQUE: Ensures that all values in a column are different.
  - CHECK: Makes sure that all values in a column satisfy certain criteria.
  - Primary Key: Used to uniquely identify a row in the table.
  - Foreign Key: Used to ensure referential integrity of the data.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

The check clause is parsed but ignored by all MySQL engines.

Create Table – Constraints -- Examples

- CREATE TABLE table_ex1 (id INTEGER NOT NULL, name varchar (10));
- CREATE TABLE employee (employee_ID integer NOT NULL, name varchar (20), pay_grade int DEFAULT 80);
- CREATE TABLE table_ex2 (id INTEGER UNIQUE, name varchar (10));
- CREATE TABLE table_ex3 (id INTEGER PRIMARY KEY, name varchar (10));
- CREATE TABLE ORDERS (Order_ID integer, Order_Date date, Customer_SID integer, Amount double,
  Primary Key (Order_ID),
  Foreign Key (Customer_SID) references CUSTOMER(SID));

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Alter Statement

- Alter statement is used to alter or change the present structure of database objects.
- Alter statement for database:

```
ALTER {DATABASE | SCHEMA} database_name
[DEFAULT] CHARACTER SET [=] charset_name
| [DEFAULT] COLLATE [=] collation_name
UPGRADE DATA DIRECTORY NAME;
```

- Alter statement for tablespace:

```
ALTER TABLESPACE tablespace_name
{ADD|DROP} DATAFILE 'file_name'
[INITIAL_SIZE [=] size]
[WAIT]
ENGINE [=] engine_name;
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Alter Table Statement

To add a column:

ALTER [ONLINE | OFFLINE] [IGNORE] TABLE *tbl_name*
ADD [COLUMN] *col_name column_definition*
[FIRST | AFTER *col_name];*

To drop a column:

ALTER [ONLINE | OFFLINE] [IGNORE] TABLE *tbl_name*
DROP [COLUMN] *col_name*

To rename a table:

ALTER [ONLINE | OFFLINE] [IGNORE] TABLE *tbl_name*
RENAME [TO] *new_tbl_name*

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Alter Table Statement

To modify column type:

ALTER [ONLINE | OFFLINE] [IGNORE] TABLE *tbl_name*
MODIFY [COLUMN] *column_name column_definition*

To add a primary key:

ALTER [ONLINE | OFFLINE] [IGNORE] TABLE *tbl_name*
ADD PRIMARY KEY (existing_column_name);

To add a partition:

ALTER [ONLINE | OFFLINE] [IGNORE] TABLE *tbl_name*
ADD PARTITION (*partition_definition);*

Infosys® © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Drop Statement

- To drop a database, i.e. to remove all the tables first and then delete the database use:

  DROP {DATABASE | SCHEMA} [IF EXISTS] *db_name;*

- To drop a table:

  DROP [TEMPORARY] TABLE [IF EXISTS] tbl_name;

- To drop a tablespace:

  DROP TABLESPACE *tablespace_name*
  ENGINE [=] *engine_name*

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Truncate Statement

- TRUNCATE TABLE command empties a table completely.
- The syntax is: TRUNCATE [TABLE] *tbl_name*
- Truncate operations drop and re-create the table, which is much faster than deleting rows one by one, particularly for large tables.
- It preserves the partitioning that is, the data and index files are dropped and re-created, while the partition definitions file is unaffected.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DDL – Rename Statement

- The syntax is:

  RENAME TABLE *tbl_name TO new_tbl_name*

- It can rename one or more tables.

- The rename operation is done atomically, which means that no other session can access any of the tables while the rename is running.

- Can be used to move a table from one database to another as follows:

  RENAME TABLE *current_db.tbl_name TO other_db.tbl_name;*

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML Statements:

Following are some of the DML statements:

- INSERT
- DELETE
- REPLACE
- UPDATE
- SELECT

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Insert Statement

- The syntax is:

  INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY]
  [IGNORE] [INTO] *table_name [(col_name,...)]*
  {VALUES | VALUE} ({*expr* | *DEFAULT*},...),(...),...
  [ON DUPLICATE KEY UPDATE *col_name=expr* [, *col_name=expr]*
  ...]

Example:

INSERT INTO table_Ex1*(id, name) VALUES(1,'Deepak');*

INSERT INTO *tbl_name () VALUES();* // *To insert null values to all columns*

Infosys® © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Insert Statement

- DELAYED: The server puts the row or rows to be inserted into a buffer, and the client issuing the INSERT DELAYED statement can then continue immediately. If the table is in use, the server holds the rows.
- LOW_PRIORITY : The execution of the INSERT is delayed until no other clients are reading from the table. Using HIGH_PRIORITY overrides the effect of the low-priority-updates and also causes concurrent inserts not to be used.
- IGNORE: It will ignore errors that occur while executing the INSERT statement and errors are treated as warnings instead.
- ON DUPLICATE KEY UPDATE: Causes a update of the row, if a duplicate value occurs for Unique / primary key value during insertion.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Insert With Select Statement

- The select statement is used with insert statement to populate the new tables with the values from one or more old tables.

- The syntax is:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
SELECT ...
[ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

- Example

```
INSERT INTO table_ex2 (id)
SELECT id
FROM table_ex1 WHERE  id > 100;
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Insert Statement

Using Set Statement: The syntax is:

INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] *table_name [(Col1, col2,..)]*
SET column1 = value1 [, column2 = value2,…]

Example:

INSERT INTO table_ex2 SET id = 100, name = 'Deepak';

Multiple insert using single insert statement:

INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] *table_name [(Col1, col2,..)]*
VALUES (v1, v2, …), (m1, m2), …

Example:

INSERT INTO TABLE_EX3 VALUES (10, 'X'), (11, 'Y');

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

INSERT INTO table_ex2 SET id = 100, name = 'Deepak';

INSERT INTO TABLE_EX3 VALUES (10, 'X'), (11, 'Y');

DML – Replace Statement

- The replace statement also used to add rows in a table. The syntax is similar to insert. The difference lies in the handling of duplicates.
- Example:

```
REPLACE INTO table_Ex1(id, name) VALUES(1,'Deepak');
REPLACE INTO table_ex2 SET id = 100, name = 'Deepak';
REPLACE INTO TABLE_EX3 VALUES (10, 'X'), (11, 'Y');
```

- IF there are duplicates rows or rows where the primary key matches with the new record, then the old records are deleted and only the new record is added.
- It is better than using delete and then insert separately.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Create table test_replace (a INT UNIQUE, b VARCHAR(2));

Replace into test_replace values (1,'x'), (2,'y'), (3,'z');

Select * from test_replace;

1	x
2	y
3	z

Replace into test_replace values (1,'x'), (2,'y'), (3,'z');

DML – Delete Statement

- The delete statement is used to delete rows from table and returns a count of the number of deleted.
- The syntax is:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
FROM table_name
[WHERE where_condition]
[LIMIT row_count]
```

- The LIMIT clause places a limit on the number of rows that can be deleted.
- Example:

```
DELETE
FROM table_ex2    WHERE  id > 100;
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Update Statement

- The update statement is used to updates columns of existing rows in the table with new values.
- The syntax is:

```
UPDATE [LOW_PRIORITY] [IGNORE]  table_name
SET col_name1={expr1} [, col_name2={expr2}] ...
[WHERE where_condition]
[ORDER BY ...]  [LIMIT row_count]
```

- The LIMIT clause places a limit on the number of rows that can be deleted.
- Example:

```
UPDATE  table_ex2
SET id = id + 1
WHERE  id > 100;
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Truncate Statement

- The truncate statement is used to delete all the rows in the table
- The syntax is:

  TRUNCATE [TABLE] table_name;

- It is similar to delete statement with no where clause.
- Considered to be faster than delete.
- Truncate returns 0 as number of rows deleted where as Delete will return the exact number of rows deleted.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Part 4 – Select Statement

Infosys © 2008, Infosys Technologies Ltd. Confidential

DML – Select Statement

SELECT is used to query the database.

```
SELECT  [ALL | DISTINCT | DISTINCTROW ]
col_name…
[FROM tablename1 [,tablename, …] ]
[WHERE condition]
[GROUP BY {col_name | expr | position}  [ASC | DESC],
[HAVING condition]
[ORDER BY {col_name | expr | position}  [ASC | DESC]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[INTO var_name [, var_name]]
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Select Statement

- Clauses must be written in the following order
  - Select
  - From
  - Where
  - Group By
  - Having
  - Order By
  - Limit

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Select Statement

- The SELECT statement is parsed as:
  - FROM
  - WHERE
  - GROUP BY
  - HAVING
  - SELECT
  - ORDER BY
- At each step the interim result set is maintained which is then further refined by the next step.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Select Statement

- **From:** Generate joins of all the tables if there are more than one tables.
- **Where:** Filter out records from the "interim result set" . If the condition matches the record is kept otherwise removed.
- **GROUP BY:** Create separate groups of rows. There may be more than one group but there is at-least one group.
- **HAVING:** Filter out all groups that don't match the conditions in the HAVING clause.
- **SELECT:** Finds out rows to be included in the final result set.
- **ORDER BY:** Sort the result set in the order specified.
- **Limit:** Constraints the number of rows returned by the SELECT statement. It takes one or two +ve integer arguments giving initial row to start with and no. of rows respectively.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Select Statement -- Examples

- **Without using FROM:**
  SELECT repeat('D',2), 3*2;
- **Selecting all the columns:**
  SELECT * from table_ex1;
- **Using order by:**
  SELECT * FROM table_ex3 Order by name;
- **Using Group by:**
  SELECT avg(id) FROM table_ex3 Group by name;
- **Using Limits:**
  SELECT * FROM table_ex3 Limit 5,10; // Display rows 6-15.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Output:

1.

repeat('D',2)	3*2
DD	6

DML – Select Statement -- WHERE vs. HAVING

- Where and Having clauses are both used to select specific records from the result set.
- Where clause is processed before the groups are created and hence it can refer to any value in the original table.
- Having clause is processed after the groups are created and hence it can only refer to aggregate information for the group.
- The Having clause cannot refer to individual columns from a table that are not also part of the group.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DML – Select Statement – Having without Where

- In the following SELECT statement:
  - There is a HAVING clause but no WHERE clause
  - The GROUP BY clause works to group several rows from the original table together to get aggregate information about the group.
  - The HAVING clause eliminates some of the resulting rows of aggregate information.

- Example:

```
SELECT department, avg(salary) as avgSal
FROM employee
GROUP BY department
HAVING avg(Salary) >=10000
ORDER BY avgSal
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

JOINS

- Joins are used when we want to retrieve information form more than one tables.
- The FROM clause will name all the tables. In MySQL 5, we can go for as big as joins for 61 tables.
- Columns to be selected can be from any table.
- If a column refers to more than one table, then it is necessary to prefix the column name with the table name.

Infosys © 2008, Infosys Technologies Ltd. Confidential

We enable you to leverage knowledge anytime, anywhere!

In relational databases, data is spread over multiple tables. Sometimes we may want data from two or more tables. A join is an operation which combines results from two or more tables.

JOINS -- Types

Following are the common types of joins available:

- Cartesian Product: Here each row from the first table is combined with all rows from the second table.
- Inner join: Allows rows from both table if and only if both tables meet the conditions specified in the Where clause.
- Equi join: Same as Inner join with the Where clause been equality condition.
- Outer join: Retrieve all rows that match the WHERE clause and also those that have a NULL value in the column used for join. Are of 2 types: Left-outer join and Right-outer join.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

JOINS -- Examples

- Cartesian Product:

  SELECT * FROM table1, table2;
  Or
  SELECT * FROM table1 JOIN table2;

- Inner join:

  SELECT * **FROM** t1, t2 **WHERE** t1.col_n1 **<op>** t2.col_n2;

  SELECT * **FROM** t1 **INNER JOIN** t2 **ON** t1.col_n1 **<op>** t2.col_n2;

  If the name of the joined column is same in both tables we can use:

  SELECT * **FROM** t1 **INNER JOIN** t2 **USING**(col1, col2, ...);

  If the operator is "=" then it is called as equi join.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

JOINS -- Examples

- Left Outer Join:

  SELECT * FROM t1 LEFT JOIN t2 ON t1.col1 <op> t2.col2;

  The left outer join displays all the rows produced by the inner join plus any row of the left table that doesn't match any row of the right table. The right table row values are shown as null.

- Right Outer Join:

  SELECT * FROM t1 RIGHT JOIN t2 ON t1.col1 <op> t2.col2;

  The right outer join displays all the rows produced by the inner join plus any row of the right table that doesn't match any row of the left table. The left table row values are shown as null.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

For details examples and queries refer the lab guide.

Part 5 – SQL Functions

Infosys © 2008, Infosys Technologies Ltd. Confidential

SQL Functions:

- Functions are invoked using parameters and they return a value.
- MySQL has a big list of SQL Function. We will discuss only few of them.
- The categories in which functions can be classified are:

1. Comparison functions
2. Mathematical functions
3. Aggregate Functions
4. String functions
5. Temporal function
6. Null related function

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

For details examples and queries refer the lab guide.

SQL Functions – Comparison Functions

- Comparison function are used to compare values.
- Some of them are as below:

1. LEAST() – Returns the smallest argument.
2. GREATEST() – Returns the largest argument.
3. IN() – Finds a value in the argument list and returns 1 if found else returns 0.
4. BETWEEN – Finds whether a value lies between two given values and return 1 if true else returns 0.
5. Interval() -- Returns number of elements less than or equal to the first element in the argument list. The list has be in sorted order after the first element.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

```
SELECT LEAST(34.0,3.0,5.0,767.0);          -- 3.0
SELECT LEAST('B','A','C');                 -- A

SELECT GREATEST(34.0,3.0,5.0,767.0);   -- 767.0
SELECT GREATEST('B','A','C');                          -- C

SELECT 2 IN (0,3,5,7);                                 -- 0
SELECT 'wefwf' IN ('wee','wefwf','weg');     -- 1
SELECT val1 FROM tbl1 WHERE val1 IN (‘a',‘b',‘c');

SELECT 2 BETWEEN 1 AND 3, 2 BETWEEN 3 and 1;    -- 1, 0
SELECT 1 BETWEEN 2 AND 3;                                        -- 0
SELECT 'b' BETWEEN 'a' AND 'c‘;                                  -- 1
SELECT val1 FROM tbl1 WHERE val1 BETWEEN 3 AND 10;
SELECT val1 FROM tbl1 WHERE val1 BETWEEN ‘A’ AND ‘Z’;

SELECT INTERVAL(7.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10);                    -- 7
```

SQL Functions – Control flow functions

- Control flow functions are used to choose between different values..
- Following are two control functions:

1. IF(): The syntax is :

```
IF(expr, valueiftrue, valueiffalse)
```

2. CASE: The syntax is:

```
CASE expr
   WHEN expr1 THEN do_this
   WHEN expr2 THEN do_this ...
   ELSE do_this
END
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

```
SELECT IF(10<5,‘True',‘False');                          -- False
SELECT IF(STRCMP('Deepak','deepak'),'True','False');   --False

SET @x = 10;
SELECT CASE @x
            WHEN 0 THEN 'X is equal to 0'
            WHEN 10 THEN 'X is equal to 10'
            WHEN 20 THEN 'X is equal to 20'
            ELSE 'x is either less than 0 or greater than 20'
      END AS X;
```

SQL Functions – Mathematical function

- Following are few of the mathematical functions:
  1. ABS() – Returns the absolute value of an integer.
  2. CEILING() -- Returns the smallest integer value not less than the argument.
  3. FLOOR() -- Returns the largest integer value not greater than the argument.
  4. MOD() – Returns the remainder.
  5. ROUND() – Round to the nearest integer.
  6. RAND() – Generates a random number.
  7. Trigonometric function -- SIN, COS, TAN etc.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

```
SELECT ABS(-5);                    -- 5
SELECT ABS(col) FROM T1;

SELECT CEILING(2.7);               --3
SELECT CEILING(2.4);               --3

SELECT FLOOR(2.7);                 --2
SELECT FLOOR(2.4);                 --2

SELECT MOD(27, 4);                 --  3
SELECT MOD(-27, 4);                --  -3
SELECT MOD(27, -4);                --  3
SELECT MOD(27.4, 4);               --  3.4

SELECT ROUND(2.7);                 -- 3
SELECT ROUND(2.4);                 -- 2
SELECT ROUND(-2.7);                --  -3
SELECT ROUND(-2.4);                --  -2


SELECT RAND();
```

SQL Functions – Aggregate function

- Aggregate functions perform summary operations on a set of values. These functions are often used with the GROUP BY clause to arrange values from a result set into groups.
- Some of these are as follows:
  1. AVG() Return the average value of the argument
  2. COUNT(DISTINCT): Return the count of a number of different values
  3. COUNT() Return a count of the number of rows returned
  4. MAX() Return the maximum value
  5. MIN() Return the minimum value
  6. SUM() Return the sum

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

SELECT cust_name, AVG(bill_amount) FROM customer_bill GROUP BY cust_name;

SELECT COUNT(DISTINCT name) FROM customer_bill;

SELECT COUNT(*) FROM customer_bill;

SELECT COUNT(DISTINCT name) FROM customer_bill;

SELECT cust_name, MIN(bill_amount), MAX(bill_amount) FROM customer_bill GROUP BY cust_name;

SELECT cust_name, SUM(bill_amount) FROM customer_bill GROUP BY cust_name;

SQL Functions – String function

- Following are some of the string functions available:
  - ASCII(): Return numeric value of left-most character.
  - CHAR_LENGTH(): Return number of char's in argument.
  - CHAR(): Return the character for the integer.
  - CONCAT_WS(): Return concatenate with separator
  - CONCAT(): Return concatenated string
  - INSERT(): Insert a substring at the specified position.
  - INSTR(): Return index of the first occurrence of substring
  - LEFT(): Return leftmost number of characters as specified
  - LENGTH(): Return the length of a string in bytes
  - LIKE: Used for simple pattern matching
  - LOCATE(): Return position of first occurrence of substring
  - LOWER() Return the argument in lowercase

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

SQL Functions – String function

- Following are some of the string functions available:
  - LPAD(): Returns left-padded new string with given string
  - LTRIM(): Remove leading spaces
  - MATCH: Perform full-text search
  - REPLACE(): Replace occurrences of a specified string
  - REVERSE(): Reverse the characters in a string
  - RIGHT(): Return specified rightmost number of characters
  - RPAD() :Append string the specified number of times
  - RTRIM(): Remove trailing spaces
  - STRCMP(): Compare two strings
  - SUBSTR(): Return the substring as specified
  - TRIM() Remove leading and trailing spaces
  - UPPER() Convert to uppercase

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

SQL Functions – Temporal functions

- Some functions related with Date & Time are as follows:
  - CURDATE() & CURTIME(): Return the current date & time.
  - DATE_FORMAT(): Format date as specified
  - DATE_SUB(): Subtract two dates
  - DAYOFMONTH(): Return the day of the month (0-31)
  - DAYOFYEAR(): Return the day of the year (1-366)
  - NOW(): Return the current date and time
  - SYSDATE(): Return the time of its call
  - WEEK() Return the week number
  - YEAR() Return the year

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

SQL Functions – NULL Related functions

- ISNULL(): Test if the argument is null or not.
- IFNULL(expr1, expr2): If the first expression is not null, it returns expr1 else returns expr2.
- NULLIF(expr1, expr2): Returns NULL if expr1 and expr2 are equal else returns expr1.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

```
SELECT IFNULL(1,0);                    -- 1
SELECT IFNULL(NULL,10);                -- 10
SELECT IFNULL(1/0,10);                 --  10
SELECT IFNULL(1/0,'yes');              -- yes
```

Part 5 – Advanced SQL

Infosys © 2008, Infosys Technologies Ltd. Confidential

We enable you to leverage knowledge anytime, anywhere!

Subqueries

- A subquery is a select statement inside another select statement.
- Types of subqueries:
  - Scalar Subqueries -- Subquery returns one row and one column.
  - Row Subqueries -- Subquery returns a single row.
  - Column Subqueries -- Subquery returns a single column
  - Table Subqueries -- Subquery returns one or more rows with one or more columns.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Subqueries:

- A subquery can contain any of the keywords or clauses that an ordinary SELECT can contain including distinct, group by , order by, limit etc.
- A subquery's outer statement must be one of: SELECT, INSERT, UPDATE, DELETE, SET, or DO.
- In a subquery, we cannot modify a table and select from the same table in a subquery.
- Simple Example:
  - SELECT ((SELECT SUM(sal) FROM personal) / (SELECT count(*) FROM employee)) AS avg_sal;
  - SELECT id FROM employee
    WHERE sal = (SELECT MAX(sal) FROM personal);

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

create table employee (id INT PRIMARY KEY, name VARCHAR(10), sal INT);

create table personal (id INT PRIMARY KEY, sal INT);

replace into employee values (1, 'a', 500), (2, ‘b', 5000), (3, ‘c', 1000), (4, ‘d', 3000), (5, ‘e', 6000);

replace into personal values (1, 500), (2, 5000), (3, 1000), (4, 3000), (5, 6000);

SELECT id FROM employee WHERE sal = (SELECT MAX(sal) FROM personal);

SELECT CONCAT(‘The emp name for the id = 1 is: , (SELECT name FROM employee WHERE id = 1)) AS output;

Correlated Subqueries:

- A non correlated subquery have no reference to outer query and can be executed as a separate query.
- A correlated subquery have reference to outer query and can not be executed as a separate query.
- Simple Example:

```
SELECT name, location
FROM employee e, personal p
WHERE e.sal = (SELECT max(sal)
               FROM personal
               WHERE e.id = p.id);
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

create table employee (id INT PRIMARY KEY, name VARCHAR(10), sal INT);

create table personal (id INT PRIMARY KEY, sal INT);

replace into employee values (1, 'a', 500), (2, ‘b', 5000), (3, ‘c', 1000), (4, ‘d', 3000), (5, ‘e', 6000);

replace into personal values (1, 500), (2, 5000), (3, 1000), (4, 3000), (5, 6000);

alter table personal add location Varchar(10);

replace into personal values (1, 1000, 'Delhi'), (2, 3000, 'Bangalore'), (3, 500, 'Chennai'), (4, 501, 'Pune'), (5, 5000, 'Hyderabad');

select name, location from employee e, personal p where e.sal = (select max(sal) from personal where e.id = p.id);

Views:

- A view is a database object that us defined in terms of a select statement that retrieves the data you want to produce. Also called as virtual tables.
- Access to data become simplified as we can select only restricted number of columns, data can be selected from multiple column.
- Views are mainly used ot give diffrent view of the table to differnt users.
- Views can help in hiding the details of actual tables. A view can be created with 2 underlying tables will give a user feel hat he is working on a single table.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

create table employee (id INT PRIMARY KEY, name VARCHAR(10), sal INT);

create table personal (id INT PRIMARY KEY, sal INT);

replace into employee values (1, 'a', 500), (2, ‘b', 5000), (3, ‘c', 1000), (4, ‘d', 3000), (5, ‘e', 6000);

replace into personal values (1, 500), (2, 5000), (3, 1000), (4, 3000), (5, 6000);

alter table personal add location Varchar(10);

replace into personal values (1, 1000, 'Delhi'), (2, 3000, 'Bangalore'), (3, 500, 'Chennai'), (4, 501, 'Pune'), (5, 5000, 'Hyderabad');

select name, location from employee e, personal p where e.sal = (select max(sal) from personal where e.id = p.id);

Views:

- Syntax:

```
Create [or replace] [algorithm = algorithm_type]
view view_name [(col list)]
as select ...
[with [cascade | local] check option]
```

- Replace clause: Will drop any existing view with the same name to be dropped prior to creating the new view.
- Algorithm: Specifies algorithm to be used when view is invoked. Can take 3 values -- Undefined, Merge or Temptable.
- Check: Ensures that any updates / changes made to the values in the view are according to the check conditions given.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

The undefined allows the MySQL to choose its own algorithm. The merge option processes a statement that refers to the view by merging parts of the view definition into corresponding parts of the statement and executing the merged statement. The temptable makes MySQL use a temp table and first retrieve the view content in the temp table and then use the temp table. This option makes a view unupdatable.

Procedures & Functions:

- Clients can create functions and procedure and store in the server for later use. These are commonly called as routines.
- Body of these routines consists of a valid SQL statement. This includes DML / DDL statements like Select / Insert / Create / Drop along with control structure statement.
- A procedure does not return a value where as a function returns a single value to the caller.
- A function can be used within an expression where as a procedure can't be invoked.
- All procedures and functions created are by default associated with a database.
- Procedures can contain SQL transaction statements such as COMMIT. Stored functions may not contain statements that perform explicit or implicit commit or rollback.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Creating Procedures:

- Syntax:

  Creating procedure:

  ```
  CREATE [DEFINER = { user_n | curr_name }]
  PROCEDURE proc_name ([parameter[,...]])
  [characteristic ...] proc_body
  ```

- Calling a Procedure:

  ```
  CALL procedure_name([parameter[,...]])
  ```

  For OUT and INOUT parameter first initialize it and then pass it to the procedure. After the execution of the procedure, check foe the new value.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

The CALL statement invokes a stored procedure that was defined previously with CREATE PROCEDURE.

Reference: MySQL tutorial at www.mysql.com

As of MySQL 5.1.13, stored procedures that take no arguments can be invoked without parentheses. That is, CALL p() and CALL

p are equivalent. CALL can pass back values to its caller using parameters that are declared as OUT or INOUT parameters. When the procedure returns,a client program can also obtain the number of rows affected for the final statement executed within the routine: At the SQL level, call the ROW_COUNT() function; from the C API, call the mysql_affected_rows() function.

To get back a value from a procedure using an OUT or INOUT parameter, pass the parameter by means of a user variable, and then

check the value of the variable after the procedure returns. (If you are calling the procedure from within another stored procedure or

function, you can also pass a routine parameter or local routine variable as an IN or INOUT parameter.) For an INOUT parameter,

initialize its value before passing it to the procedure.

Procedures Parameters:

- There are 3 types of parameters that can be defined when creating procedure: IN, OUT and INOUT.
- IN: This type of parameter is used to pass a value into the procedure. The modification to the value of this parameter is not visible outside. All parameters by default is IN unless specified explicitly.
- OUT: This type of parameter is used to pass a value to the caller Its initial value is NULL and the modified value is visible to the caller.
- INOUT: This type of parameter is combination of IN and OUT.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Example:

```
Delimiter //
Create procedure ptest(IN vin INT, OUT vout INT, INOUT vinout INT)
BEGIN
        Select vin,vout, vinout;
        Set vin = 1; vout=0, vinout=5;
END;
//
```

Procedures : Example

- CREATE PROCEDURE emp_sal(IN eid INT)
  BEGIN
  SELECT sal FROM employee
  WHERE id = eid;
  END;
- CREATE PROCEDURE emp_sal (IN eid INT, OUT s INT)
  BEGIN
  SELECT sal INTO s FROM employee
  WHERE id = eid;
  END;
- Calling the emp_sal procedure:
  - SET @s = -1;
  - SET @eid = 1000;
  - CALL emp_sal (@eid);
  - CALL emp_sal (@eid, @s);
  - SELECT @s;

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Required to change the delimiter:

```
Delimiter //     [this could be any special character but be careful in choosing it]
CREATE PROCEDURE emp_sal(IN eid INT)
BEGIN
        SELECT sal  FROM employee WHERE id = eid;
END;
//
```

Creating Functions:

- Syntax:

  Creating Function

  ```
  CREATE
  [DEFINER = { user | CURRENT_USER }]
  FUNCTION function_name ([parameter[,...]])
  RETURNS data_type
  [characteristic ...] func_body
  ```

- The call syntax cannot be used for calling function.
- Functions have a return type and the return values must be handled.
- The parameters are all of IN type. The output is given by the returns clause.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Functions : Example

- CREATE FUNCTION area_circle (rad INT)
  RETURNS float
  BEGIN
  Return pi() * rad * rad;
  END;
- CREATE FUNCTION area_rect (len INT, breath INT)
  RETURNS int
  BEGIN
  Return len * breath;
  END;
- Calling the emp_sal procedure:
  - Select area_circle(5);
  - Select area_rect(5,6);

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Compound Statement in Procedure & Function:

- There could be more than one statement in the body of the routines.
- The structure of the routine body is:

```
[DECLARE var1, var2, …
  SET var1 = val1, var2 = val2, …]
BEGIN
      …
      [BEGIN
        …
      END;]
END;
```

- The routines can have multiple nested BEGIN and END statement.
- Declare section is used to declare variables required for local calculation. SET is required to give values to the variables.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Declare Statement : Example

```
delimiter //

CREATE FUNCTION  emp_sal_bonus (eid INT, bonus INT)
RETURNS FLOAT
    BEGIN
        Declare   old_sal INT;
        Declare   new_sal float;
        SET new_sal=0.0;
        SELECT sal  into old_sal FROM employee
        WHERE id =  eid;
        SET new_sal = (old_sal + old_sal * (bonus / 100));
        RETURN new_sal;
    END;
//
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Compound Statement: Cursor

- Cursors help in accessing one row at a time.
- Cursors temporary stores the result set returned by a select clause.
- Cursors are read only and cannot modify the underlying table.
- Cursors are supported inside stored procedures, functions and triggers.
- Cursors are non scrollable i.e. cursors can be traversed only in one direction and no rows can be skipped.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Compound Statement: Cursor

- To use a cursor we declare it using DECLARE CURSOR statement and associated a select clause with it.

  DECLARE cur_name CURSOR FOR select_stat;

- To open a cursor use

  OPEN cursor_name;

- To fetch next row from the cursor into variables use:

  FETCH cur_name INTO var1 [, var2]...

- To close a cursor use:

  CLOSE cursor_name;

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Compound Statement: Control Statements

There are 2 categories of control statement:

- Testing Conditions:
  - IF
  - CASE (done in previous slides)
- Iteration:
  - LOOP
  - REPEAT
  - WHILE

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Control Statements: Testing Conditions

```
IF expression
   THEN statement
   [ELSEIF expression THEN statement]...
   [ELSE statement]
ENDIF


CASE expr
   WHEN expression THEN statement
   WHEN expression THEN statement ...
   ELSE statement
END
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Control Statements: Iteration

- Loop constructs an unconditional loop with no termination. It is required explicitly write a terminating condition inside the loop.

```
LOOP
    statement1
    ...
END LOOP
```

- Exit from the loop can be done by using leave statement or Return statement in case of function.
- To do the loop again use ITERATE statement. Use labels to mark the loops and use the iterate and leave statement.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

```
Create procedure iteration_ex
Begin
        DECLARE i INT DEFAULT o;
        l1: LOOP
                SET i = i + 1;
                IF i < 10 THEN ITERATE l1;
                ELSEIF i>20 THEN LEAVE l1;
                END IF;
                SELECT 'I is between 10 and 20';
      END LOOP l1;
END;
```

Control Statements: Iteration

- The repeat statement creates a conditional loop.

```
REPEAT
    statement1
    ...
UNTIL expression
END REPEAT;
```

- The while statement also creates a conditional loop but the condition appears in the beginning.

```
WHILE expression DO
    statement1
    ...
END WHILE;
```

- WHILE may not even execute once but the repeat loop will execute at-least one time.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

```
Create procedure iteration_ex
Begin
        DECLARE i INT DEFAULT o;
        l1: LOOP
                SET i = i + 1;
                IF i < 10 THEN ITERATE l1;
                ELSEIF i>20 THEN LEAVE l1;
                END IF;
                SELECT 'I is between 10 and 20';
      END LOOP l1;
END;
```

Control Statements: Iteration Example

- REPEAT example:

```
Create procedure iteration_ex
Begin
    DECLARE i INT DEFAULT 0;
    REPEAT
      SET i = i + 1;
      SELECT 'I is greater than
                        5';
    UNTILL i >5;
    END REPEAT:
END;
```

- WHILE example:

```
Create procedure iteration_ex
Begin
    DECLARE i INT DEFAULT 0;
    WHILE i < = 5 DO
      SET i = i + 1;
      SELECT 'I is greater than
                  5';
    UNTILL i >5;
END;
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

```
Create procedure iteration_ex
Begin
        DECLARE i INT DEFAULT o;
        l1: LOOP
                SET i = i + 1;
                IF i < 10 THEN ITERATE l1;
                ELSEIF i>20 THEN LEAVE l1;
                END IF;
                SELECT 'I is between 10 and 20';
      END LOOP l1;
END;
```

Example using Cursor:

```
CREATE PROCEDURE cursordemo ()
BEGIN
    DECLARE i, s INT;
    DECLARE cursor1 CURSOR FOR SELECT id, sal FROM
    employee;
    OPEN cursor1;
    REPEAT
        FETCH cursor1 INTO i, s;
        IF s < 1000 THEN
                UPDATE employee SET sal = sal + 1000 WHERE id = i;
        ELSE
                UPDATE employee SET sal = sal + 500 WHERE id = i;
        END IF;
    UNTIL i >=5
    END REPEAT;
    CLOSE cursor1;
END;
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Drop Procedure and Functions:

- To drop a procedure use:

  DROP PROCEDURE [IF EXISTS] proc_name;

- To drop a function use:

  DROP FUNCTION [IF EXISTS] func_name;

- To alter a procedure / function body first drop it and then recreate it. Cannot use the ALTER statement to do the same.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

DROP PROCEDURE cursordemo;

Stored Routine Metadata:

- The INFORMATION_SCHEMA database has a ROUTINES table that contains information about the stored routines.

```
SELECT * FROM INFORMATION_SCHEMA.ROUTINES WHERE
ROUTINE_NAME = 'ANY NAME';
```

- SHOW statements can also be used for the same purpose.

```
SHOW PROCEDURE STATUS LIKE 'pattern';
SHOW FUNCTION STATUS;
```

- To display the definition of an individual stored procedure or function use:

```
SHOW CREATE PROCEDURE <proc_name>;
SHOW CREATE FUNCTION <func_name>;
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

SELECT * FROM INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME = ‘cursordemo';

SHOW CREATE PROCEDURE cursordemo;

Transaction Control Statement:

- Following are the transaction control statement commonly used:
  - START TRANSACTION – Starts a transaction.
  - COMMIT -- Saves the changes made in the database.
  - SET autocommit = {0 | 1} – By default the values is 1 and it is enabled. Can change to 0 if want to disable.
  - ROLLBACK – undo the changes made by the current transaction.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

```
START TRANSACTION;
SELECT @x:=sum(sal) FROM employee;
UPDATE empl_sum SET sum_sal=@x;
COMMIT;
```

Savepoints are there for only innodb engine.

Triggers:

- A trigger is an object that belongs to the database.
- A trigger defines a set of action and can be activated when a specified SQL operation like a DELETE, INSERT or UPDATE, occurs on a table.
- A trigger can be used for
  - Validate of transitional input data. Considered to be better than check constraints.
  - Automatically generating new values for inserting rows.
  - Read and Write to other tables.
  - Support alerts (Ex: e-mail messages).

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

As triggers belongs to database so each trigger belonging to any table must have unique name across database.

Transitional data: Validations that require comparisons between the value before and after an update operation.

Reading from other tables are for cross-referencing purposes where as writing to other tables is for audit-trail purposes.

Triggers:

- A trigger gets activated (executes) on the occurrence of a particular event.
- One trigger can be defined only for one type of event i.e. INSERT / UPDATE / DELETE on a table, but we can defined multiple trigger on the same table.
- Triggers can be activated either before or after the event i.e. there can be two triggers per event.
- There cannot be two triggers for a given table that have the same trigger action time and event.
- A trigger must always refer to a permanent table.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

As triggers belongs to database so each trigger belonging to any table must have unique name across database.

Transitional data: Validations that require comparisons between the value before and after an update operation.

Reading from other tables are for cross-referencing purposes where as writing to other tables is for audit-trail purposes.

Triggers:

- Syntax

```
CREATE TRIGGER trig_name
{BEFORE | AFTER}
{INSERT | UPDATE | DELETE}
ON table FOR EACH ROW
statements...
```

- In the trigger definition, we can refer to columns of the rows been inserted / deleted or updated by using the prefix OLD / NEW accordingly.
  - For insert OLD is not allowable, can refer new values.
  - For delete NEW is not allowable, can refer old values.
  - For update NEW and OLD both are allowed.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

To refer old value of a column, prefix OLD. with the column name.

To refer new value of a column, prefix NEW. with the column name.

Triggers Example:

- Example: Consider the tables in the notes page:

```
Delimiter //

CREATE TRIGGER trig_ex
BEFORE INSERT ON one FOR EACH ROW
BEGIN
INSERT INTO two SET pid = NEW.eid;
DELETE FROM three WHERE xid = NEW.eid;
UPDATE four SET v1 = v1 + 1 WHERE mid = NEW.eid;
END;
//
```

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

```
CREATE TABLE one(eid INT);
CREATE TABLE two(pid INT);
CREATE TABLE three (xid INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
CREATE TABLE four(mid INT NOT NULL AUTO_INCREMENT PRIMARY KEY, v1 INT DEFAULT 0);
```

INSERT INTO three VALUES (NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL);

INSERT INTO four (mid) VALUES (0), (0), (0), (0), (0), (0), (0), (0), (0), (0);

INSERT INTO one VALUES (1), (3), (1), (7), (1), (8), (4), (4);

Metadata Information:

- Information about the structure of the database is called metadata.
- Metadata involves getting information about databases, tables, columns, indexes and stored routines etc.
- Following are some ways to access metadata:
  - mysqlshow
  - SHOW statement
  - Using INFORMATION_SCHEMA

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

As triggers belongs to database so each trigger belonging to any table must have unique name across database.

Transitional data: Validations that require comparisons between the value before and after an update operation.

Reading from other tables are for cross-referencing purposes where as writing to other tables is for audit-trail purposes.

Metadata Information: SHOW statement

- SHOW DATABASES [LIKE 'pattern']
  List the names of all available databases.
- SHOW TABLES [FROM database_name]
  Lists the tables in the default database or from the database name given in from clause.
- SHOW [FULL] COLUMNS FROM table_name [WHERE condition]
  Displays (full) column structure information for the table.
- DESCRIBE table_name
  Is same as show column statement.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Metadata Information: mysqlshow statement

mysqlshow is a client program that produces metadata about databases and tables.

- mysqlshow [options]
  List the names of all available databases. The options are --host / --user / --help etc.
- mysqlshow [options] <database_name>
  List the names of all tables available in the database.
- mysqlshow [options] <database_name> <table_name>
  Displays column structure information for the table.
- mysqlshow [opt] <db_name> <table_name> <column_name>
  Displays full column structure information for only the column name.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Metadata : INFORMATION_SCHEMA

- INFORMATION_SCHEMA is a database having metadata for database.
- Not stored, contains tables having all information.
- To view list of tables in the INFORMATION_SCHEMA database use:

```
SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'INFORMATION_SCHEMA';
```

- Following are some of the tables:

COLUMNS	ENGINES	PARTITIONS
ROUTINES	TABLES	TABLE_CONSTRAINTS
TRIGGERS	VIEWS	

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Metadata : INFORMATION_SCHEMA

- To display the names of the columns in a given information_schema table use:

```
SELECT column_name
FROM information_schema.columns
WHERE table_schema = 'information_schema'
        and table_name = 'routines';
```

- Can use all the select clause for getting specific information.

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Debugging:

Following are some statements that help in understanding the mysql errors / warnings:

- SHOW WARNINGS -- It displays the reasons for the warnings that occurred in the last executed statement.
- SHOW ERRORS -- Similar to above, but only display messages for that error.
- PERROR -- shows details of the error that occurred at the OS level. For perror we need the errno displayed at the execution time.

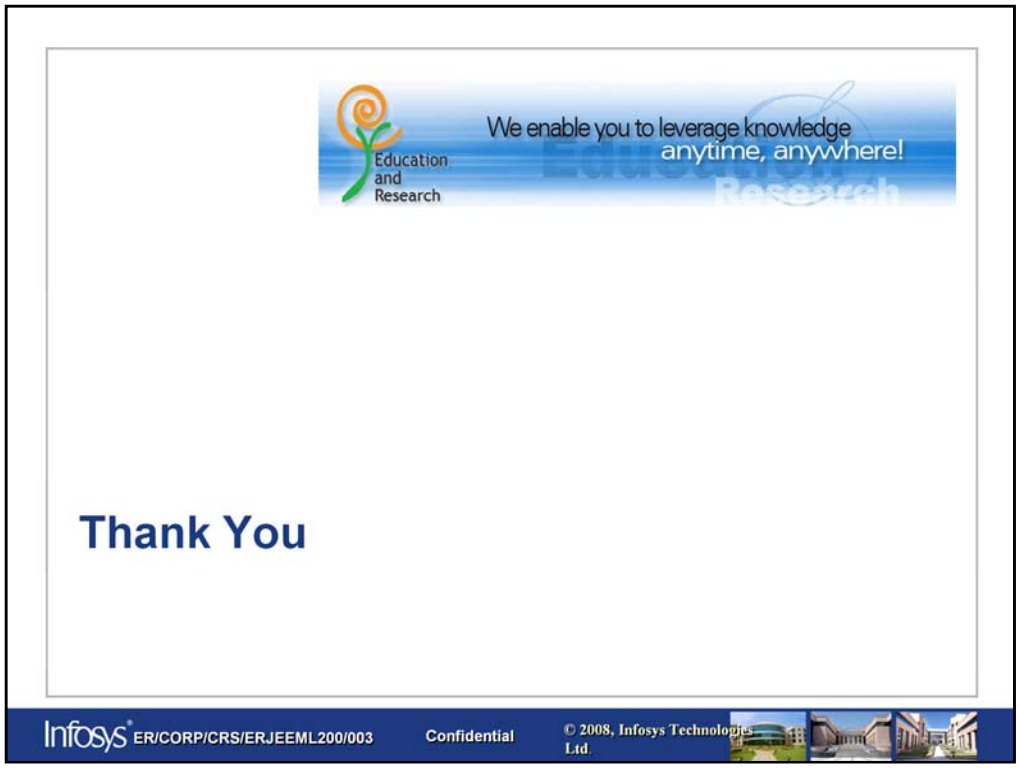
Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

Summary

- Introduction to DBMS
- Introduction to MySQL & MySQL Product Suite
- MySQL Server and Client
- MySQL Architecture
- Storage Engines
- SQL Basics, SQL Statements: DDL, DML, Transaction Control statement, Select Statement, Joins, SQL Functions
- Advanced SQL -- Subqueries, Views, Procedures & Functions, Cursors.
- Triggers
- Metadata information & Debugging

Infosys © 2008, Infosys Technologies Ltd. Confidential We enable you to leverage knowledge anytime, anywhere!

The plan may change.



Make sure that the copyright notice is present in this first slide. It is inserted as a text box in this slide only and not part of 'Slide Master'.

After replacing <Course Name> placeholder above with the appropriate course name, make sure to fill the relevant information in 'File Properties' dialog box also (File Menu -> Properties)