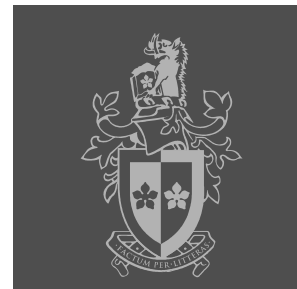**HIT6323/3323 – Web Programming**

Module 07 – Databases and MySQL

Delivered by Dr. Jinjun Chen
(jchen@ict.swin.edu.au)

Faculty of ICT

Outline

Working with Databases and MySQL

- Study the basics of databases and MySQL

- Work with MySQL databases

- Manage user accounts

- Define database tables

- Work with database records

Reading: Textbook Chapter 8

Introduction to Databases

- A **database** is an ordered collection of information from which a computer program can quickly access information

- Each row in a database table is called a record

- A **record** in a database is a single complete set of related information

- Each column in a database table is called a field

- **Fields** are the individual categories of information stored in a record

---

Introduction to Databases (continued)

Rows				Fields	
last_name	first_name	address	city	state	zip
Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Figure 8-1 Employee directory database

- A **flat-file database** stores information in a single table

- A **relational database** stores information across multiple related tables

Understanding Relational Databases

- Relational databases consist of one or more related tables

- A **primary table** is the main table in a relationship that is referenced by another table

- A **related table** (or "child table") references a primary table in a relational database

Understanding Relational Databases (continued)

- A **primary key** is a field that contains a unique identifier for each record in a primary table

- A **primary key** is a type of index, which identifies records in a database to make retrievals and sorting faster

- A **foreign key** is a field in a related table that refers to the primary key in a primary table

- Primary and foreign keys link records across multiple tables in a relational database

One-to-One Relationships

- A **one-to-one relationship** exists between two tables when a related table contains exactly one record for each record in the primary table

- Create one-to-one relationships to break information into multiple, logical sets

- Information in the tables in a one-to-one relationship can be placed within a single table

- Make the information in one of the tables confidential and accessible only by certain individuals

---

One-to-One Relationships (continued)

Primary key

Employees table

employee_id	last_name	first_name	address	city	state	zip
101	Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
102	Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
103	Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
104	Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
105	Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Foreign key

Payroll table

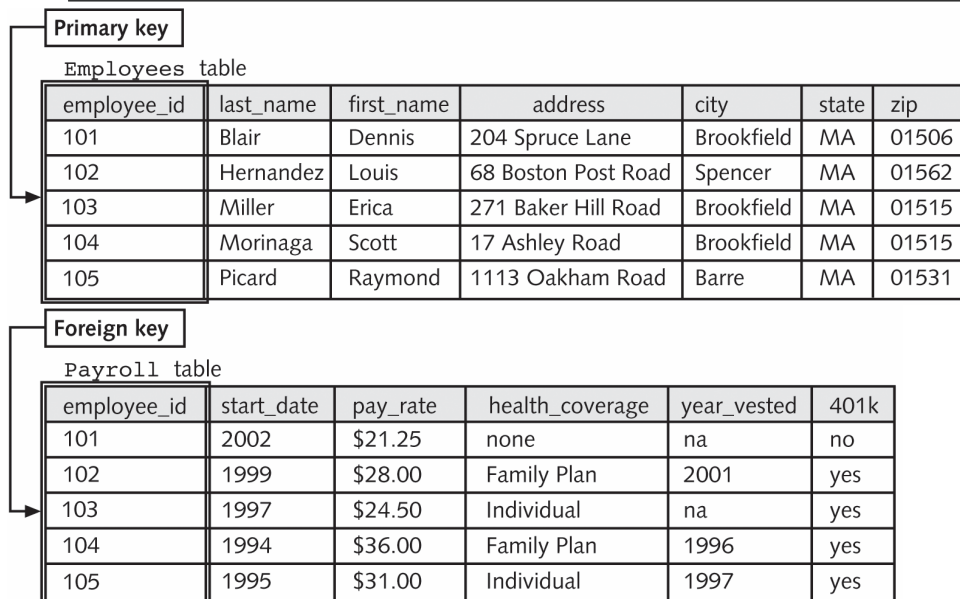| employee_id	start_date	pay_rate	health_coverage	year_vested	401k
101	2002	$21.25	none	na	no
102	1999	$28.00	Family Plan	2001	yes
103	1997	$24.50	Individual	na	yes
104	1994	$36.00	Family Plan	1996	yes
105	1995	$31.00	Individual	1997	yes

Figure 8-2 One-to-one relationship

One-to-Many Relationship

■ A **one-to-many relationship** exists in a relational database when one record in a primary table has many related records in a related table

■ Breaking tables into multiple related tables to reduce redundant and duplicate information is called **normalization**

■ Provides a more efficient and less redundant method of storing this information in a database

One-to-Many Relationship (continued)

employee_id	last_name	first_name	language
101	Blair	Dennis	JavaScript
101	Blair	Dennis	ASP.NET
102	Hernandez	Louis	JavaScript
102	Hernandez	Louis	ASP.NET
102	Hernandez	Louis	Java
103	Miller	Erica	JavaScript
103	Miller	Erica	ASP.NET
103	Miller	Erica	Java
103	Miller	Erica	C++
104	Morinaga	Scott	JavaScript
104	Morinaga	Scott	ASP.NET
104	Morinaga	Scott	Java
105	Picard	Raymond	JavaScript
105	Picard	Raymond	ASP.NET

Figure 8-3 Table with redundant information

One-to-Many Relationship (continued)

Figure 8-4 One-to-many relationship

Employees table

employee_id	last_name	first_name	address	city	state	zip
101	Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
102	Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
103	Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
104	Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
105	Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Languages table ("many" side)

employee_id	language
101	JavaScript
101	ASP.NET
102	JavaScript
102	ASP.NET
102	Java
103	JavaScript
103	ASP.NET
103	Java
103	C++
104	JavaScript
104	ASP.NET
104	Java

One record on the top table is linked to many records in the bottom table

---

Many-to-Many Relationship

- A **many-to-many relationship** exists in a relational database when many records in one table are related to many records in another table
  e.g. relationship between programmers and languages

- Must use a **junction table** which creates a one-to-many relationship for each of the two tables in a many-to-many relationship

- A junction table contains foreign keys from the two tables

Many-to-Many Relationship (continued)

Employees table

employee_id	last_name	first_name	address	city	state	zip
101	Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
102	Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
103	Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
104	Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
105	Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Languages table

language_id	language
10	JavaScript
11	ASP.NET
12	Java
13	C++

Experience junction table

employee_id	language_id	years
101	10	5
101	11	4
102	10	3
102	11	2
102	12	3
103	10	2
103	11	3
103	12	6
103	13	3
104	10	7
104	11	5

Figure 8-5 Many-to-many relationship

Working with Database Management Systems

- A **database management system** (or DBMS) is an application or collection of applications used to access and manage a database

- A **schema** is the structure of a database including its tables, fields, and relationships

- A **flat-file database management system** is a system that stores data in a flat-file format

- A **relational database management system** (or RDBMS) is a system that stores data in a relational format

Examples of RDBMS

Working with Database Management Systems (continued)

Important aspects of database management systems:

- The structuring and preservation of the database file

- Ensuring that data is stored correctly in a database's tables, regardless of the database format

- Querying capability

- (also security)

Working with Database Management Systems (continued)

- A **query** is a structured set of instructions and criteria for retrieving, adding, modifying, and deleting database information

- **Structured query language** (or SQL – pronounced as sequel) is a standard data manipulation language used among many database management systems

- **Open database connectivity** (or ODBC) allows ODBC-compliant applications to access any data source for which there is an ODBC driver

Querying Databases with Structured Query Language

Table 8-1 Common SQL keywords

Keyword	Description
DELETE	Deletes a row from a table
FROM	Specifies the tables from which to retrieve or delete records
INSERT	Inserts a new row into a table
INTO	Determines the table into which records should be inserted
ORDER BY	Sorts the records returned from a table
SELECT	Returns information from a table
UPDATE	Saves changes to fields in a record
WHERE	Specifies the conditions that must be met for records to be returned from a query

e.g. `select * from Employees`

Logging in to MySQL

- (MySQL is not only open source, but also fast and reliable)
- Enter the following command:

  ```
  mysql –h host –u user –p
  Mysql –u user -p
  ```

- Two accounts are created:
  - Anonymous user account allows login without specifying a username or password (Note: security issue)
  - `root` account (the primary administrative account for MySQL) is created without a password

    ```
    mysql –u root
    ```

- Log out with the `exit` or `quit` commands

Logging in to MySQL (continued)

```
C:\Program Files\MySQL\MySQL Server 4.1\bin>mysql -u dongosselin -p
Enter password: **********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6611 to server version: 4.1.9-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```
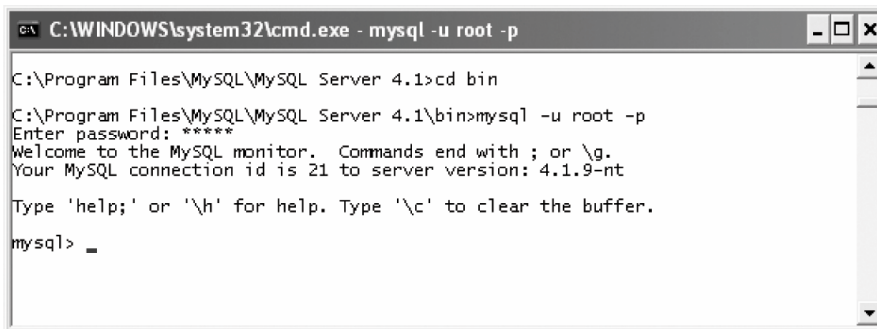

Figure 8-8 MySQL Monitor on a Windows platform

Working with the MySQL Monitor

- At the `mysql>` command prompt terminate the command with a semicolon

    ```
    mysql> SELECT * FROM inventory;
    ```

- Without a semicolon, the MySQL Monitor enters a multiple-line command and changes the prompt to ->

    ```
    mysql> SELECT * FROM inventory
        -> WHERE make = "washburn";
    ```

- The SQL keywords entered in the MySQL Monitor are not case sensitive

Understanding MySQL Identifiers

- Identifies for databases, tables, fields, indexes, and aliases

- An **alias** is an alternate name used to refer to a table or field in SQL statements

- The case sensitivity of database and table identifiers depends on the operating system
  - □ Not case sensitive on Windows platforms
  - □ Case sensitive on UNIX/Linux systems

- MySQL stores each database in a directory of the same name as the database identifier

- Field and index identifiers are case insensitive on all platforms

Getting Help with MySQL Commands


```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

For info on MySQL books, utilities, consultants, etc., visit:
    http://www.mysql.com/portal

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear command.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set query delimiter.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
notee     (\t) Don't write into outfile.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute a SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
tee       (\T) Set outfile [to_outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.

For server side help, type 'help contents'

mysql> _
```

Figure 8-9 MySQL command help

Selecting a Database

- Use the `SHOW DATABASES` statement to view the databases that are available

- Use the `USE DATABASE` statement to select the database to work with

- Use the `SELECT DATABASE()` statement to display the name of the currently selected database

- The `mysql` database is installed to contain user accounts and information that is required for installation of the MySQL database server

- The `test` database is installed to ensure that the database server is working properly

Selecting a Database (continued)


```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 26 to server version: 4.1.9-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SHOW DATABASES;
+--------------+
| Database     |
+--------------+
| test         |
| mysql        |
+--------------+
2 rows in set (0.01 sec)

mysql> use mysql;
Database changed
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| mysql      |
+------------+
1 row in set (0.00 sec)

mysql> _
```

Figure 8-10 MySQL Monitor after selecting a database

Creating Databases

- Use the `CREATE DATABASE` statement to create a new database:

```
mysql> CREATE DATABASE guitars;
Query OK, 1 row affected (0.02 sec)
```

- (To use a new database, select it by executing the `use database` statement)

- Before adding records to a new database, first define the tables and fields that will store the data

Deleting Databases

- Use the `DROP DATABASE` statement to remove all tables from the database and to delete the database

- The syntax for the `DROP DATABASE` statement is:

```
DROP DATABASE database;
```

- You must be logged in as the root user or have `DROP` privileges to delete a database

Securing the Initial MySQL Accounts

- Deleting the Anonymous User Account

```
mysql> DELETE FROM mysql.user WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

- Assigning a Password to the Root Account

```
mysql> UPDATE mysql.user SET Password = PASSWORD('newpwd')
    -> WHERE User = 'root';
mysql> FLUSH PRIVILEGES;
```

- The password assigned to the root account and other user accounts is case sensitive

Creating Users

- A **proxy** is someone or something that acts or performs a request for another person

- Create a separate account for each Web application that needs to access a database

- Use a `GRANT` statement to create user accounts and assign privileges

- **Privileges** are the operations that a user can perform with a database

Creating Users (continued)

Privilege	Description
`ALL`	Assigns all privileges to the user
`CREATE`	Allows the user to create databases, tables, and indexes
`DROP`	Allows the user to delete databases and tables
`ALTER`	Allows the user to modify table structure
`DELETE`	Allows the user to delete records
`INDEX`	Allows the user to create and delete indexes
`INSERT`	Allows the user to add records
`SELECT`	Allows the user to select records
`UPDATE`	Allows the user to modify records
`USAGE`	Creates a user with no privileges

Table 8-2 Common MySQL database privileges

`GRANT` Statement

- The syntax for the `GRANT` statement is:

```
GRANT privilege [(column)] [, privilege [(columns)]] ...
    ON {table | * | *.* | database.*}
    TO user [IDENTIFIED BY 'password'];
```

- The `GRANT` statement creates the user account
  if it does not exist and assigns the specified privileges

- If the user account already exists, the `GRANT` statement just
  updates the privileges

Revoking Privileges

- The syntax for the `REVOKE` statement is:

```
REVOKE privilege [(column)] [, privilege [(columns)]] ...
    ON {table | * | *.* | database.*}
    FROM user;
```

- The `REVOKE ALL PRIVILEGES` statement removes all privileges from a user account for a specified table or database

- You must be logged in with the `root` account or have sufficient privileges to revoke privileges from another user account

Deleting Users

- To delete a user, you must first revoke all privileges assigned to the user account for all databases
  - ☐ Use the `REVOKE ALL PRIVILEGES` statement
  - ☐ View the privileges assigned to a user account with the `SHOW GRANTS FOR user` statement

- To delete an existing user, use the `DROP USER` statement

- Use the `DROP USER user` statement to delete the account from the user table in the `mysql` database

Specifying Field Data Types

Type	Range	Storage
BOOL	-128 to 127; 0 is considered false	1 byte
INT or INTEGER	-2147483648 to 2147483647	4 bytes
FLOAT	-3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38	4 bytes
DOUBLE	-1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308	8 bytes
DATE	'1000-01-01' to '9999-12-31'	Varies
TIME	'-838:59:59' to '838:59:59'	Varies
CHAR(m)	Fixed length string between 0 to 255 characters	Number of bytes specified by m
VARCHAR(m)	Variable length string between 1 to 65,535 characters	Varies according to the number of bytes specified by m

Table 8-3 Common MySQL data types

Creating Tables

- The CREATE TABLE statement specifies the table and column names and the data type for each column

- The syntax for the CREATE TABLE statement is:

  ```
  CREATE TABLE table_name (column_name TYPE, ...);
  ```

- Execute the USE statement to select a database before executing the CREATE TABLE statement

Deleting Tables

- The `DROP TABLE` statement removes all data and the table definition

- The syntax for the `DROP TABLE` statement is:

    ```
    DROP TABLE table;
    ```

- You must be logged in as the `root` user or have `DROP` privileges to delete a table

---

Adding Records

- Use the `INSERT` statement to add individual records to a table

- The syntax for the `INSERT` statement is:

    ```
    INSERT INTO table_name VALUES(value1, value2, ...);
    ```

- The values entered in the `VALUES` list must be in the same order in which you defined the table fields

- Specify `NULL` in any fields for which you do not have a value

- Add multiple records, use the `LOAD DATA` statement

    ```
    LOAD DATA LOCAL INFILE 'file_path_name' INTO TABLE
    table_name;
    ```

Retrieving Records

- Use the `SELECT` statement to retrieve records from a table:

  `SELECT criteria FROM table_name;`

- Use the asterisk (*) wildcard with the `SELECT` statement to retrieve all fields from a table

- To return multiple fields, separate field names with a comma

  `mysql> SELECT model, quantity FROM inventory;`

Sorting Query Results

- Use the `ORDER BY` keyword with the `SELECT` statement to perform an alphanumeric sort of the results returned from a query

`mysql> SELECT make, model FROM inventory ORDER BY make, model;`

- To perform a reverse sort, add the `DESC` keyword after the name of the field by which you want to perform the sort

`mysql> SELECT make, model FROM inventory ORDER BY make DESC, model;`

Filtering Query Results

- The **criteria** portion of the `SELECT` statement determines which fields to retrieve from a table

- You can also specify which records to return by using the `WHERE` keyword

  ```
  mysql> SELECT * FROM inventory WHERE make='Martin';
  ```

- Use the keywords `AND` and `OR` to specify more detailed conditions about the records you want to return

  ```
  mysql> SELECT * FROM inventory WHERE make='Washburn'
      -> AND price<400;
  ```

Updating Records

- To update records in a table, use the `UPDATE` statement

- The syntax for the `UPDATE` statement is:

  ```
  UPDATE table_name
  SET column_name=value
  WHERE condition;
  ```

  - ☐ The `UPDATE` keyword specifies the name of the table to update

  - ☐ The `SET` keyword specifies the value to assign to the fields in the records that match the condition in the `WHERE` keyword

Deleting Records

- Use the `DELETE` statement to delete records in a table

- The syntax for the `DELETE` statement is:

  ```
  DELETE FROM table_name
  WHERE condition;
  ```

- The `DELETE` statement deletes all records that match the condition

- To delete all the records in a table, leave off the `WHERE` keyword

Summary

- A database is an ordered collection of information from which a computer program can quickly access information

- There are three basic types of relationships within a relational database: one-to-one, one-to- many, and many-to-many

- A database management system (or DBMS) is an application or collection of applications used to access and manage a database

Summary (continued)

- Structured query language (or SQL) is a standard data manipulation language used among many database management systems

- The case sensitivity of database and table identifiers depends on the operating system

- When you first install MySQL, two databases are installed: `mysql` and `test`

- You must be logged in as the root user or have `DROP` privileges to delete a database

Summary (continued)

- A proxy is someone or something that acts or performs a request for another person

- Privileges are the operations that a user can perform with a database

- You must be logged in with the `root` account or have sufficient privileges to revoke privileges from another user account

- You can specify which records to return from a database by using the `WHERE` keyword