1. Relational Database Development

152-156

Structured Query Language (SQL)

| Notes | Activity |
| --- | --- |
| Text References* + [SQL](#Basics) Pages 61 – 62
	+ [MySQL](#MySQL) Pages 62 – 66
	+ [Creating a Database](#Create) Pages 66 – 67
	+ [List Databases](#ShowDB) Pages
	+ [Delete a Database](#DropDB) Pages
	+ [Creating a Table](#CreateTable) Pages 67 – 68  81 – 85
	+ [Data Types](#DataTypes) Pages 72 – 73
	+ [Designating Foreign Keys](#foreignkey) Pages
	+ [Automatically Setting Last Modified](#lastmodified) Pages
	+ [List Tables](#ShowTables) Page 201
	+ [Show Table Structure](#ShowStructure) Page 85
	+ [Delete a Table](#DropTable) Page 71
	+ [Adding Data](#AddData) Pages 75 - 77 81 – 85
	+ [Viewing Records](#View) Pages 77 – 78
	+ [Deleting Records](#DelRec) Pages 166 – 167
	+ [Correcting Data Errors (Update)](#Update) Pages 79 – 80  168 – 172

  |  |
| 1. SQL
	* The Structured Query Language (SQL) is a semi-standard language for manipulating data in relational database.
	* In this class we will submit SQL commands to *MySQL*, an open-source SQL development environment.
 |  |
| 1. Get Your Own MySQL
	* We’ll be running MySQL and MySQL Query Browser within PortableApps. If you have not done so already, complete the [PortableApps lab](http://volkergaul.com/MSTC/Courses/Web%20Programming/Assignments/PortableApps%20Lab.pdf) (including the Installing MySQL component).
	* Though we won’t be doing this, you can install MySQL on your desktop.
		+ [dev.mysql.com](http://dev.mysql.com/)
 | MySQL |
| 1. Documentation StandardsIn these notes:
	* SQL keywords will appear in all caps, bold and blue
		+ In SQL, keywords do not have to be capitalized. They are capitalized in these notes for clarity.
	* Table names will appear with the *tbl* prefix, black
		+ Note: MySQL automatically converts all database and table names to lowercase
	* Field names will appear in camelBack notation
	* Text entered in italics must be replaced with actual table, field names
	* Objects in [square brackets] are optional.
 |  |
| 1. Creating a Database
	* MySQL requires that a database exist before you can add tables to it.
	* MySQL provides the Create Database command to do this.**CREATE DATABASE** *databasename***;**
	* MySQL Query Browser will not display the new database in the Schemas window until you refresh (right-click in Schemas)
	* All databases are stored in:E:\XAMPP\MySQL\Data
		+ C:\Program Files\MySQL\MySQL Server 6.0\Data if you’ve installed on the desktop.
		+ Each database is stored in its own folder.
		+ Unfortunately, you can NOT simply copy these folders to transfer the database to a new location.
		+ See the [Exporting and Importing MySQL Databases](http://volkergaul.com/MSTC/Courses/Relational%20Database%20Development/Handouts/Exporting%20and%20Importing%20MySQL%20databases.pdf) handout on my website for information on how to transfer a database from one location to another.
 | Create the Premiere Products and Henry Books databases. |
| * + Once you have created databases you can display a list of the available databases by entering:**SHOW DATABASES;**
		- MySQL comes with some databases.
			* Test is for experimenting with MySQL. Feel free to delete (drop) this database if you wish.
			* cdcol is a CD collection database (very small). Feel free to delete (drop) this databases if you wish.
			* The other databases are used by the MySQL server itself to track user and databases. DO NOT delete these.
 |  |
| * + If you no longer need a database you can delete it by entering:**DROP DATABASE** *databasename*;
		- There is no recovery for this command. Make sure you enter the correct database name.
		- Again, you’ll need to refresh the Query Browser Schemas window before the database disappears.
 | Drop premiereproducts and henrybooksCreate database *alexamara* |
| * + You can also create and drop databases (Schemas) using the Schemas pane in Query Browser
		- Right-click in the Schemas pane and choose Create New Schemas to create a new database.
		- Right-click any database object and choose Drop Schemas to delete the database.
 |  |
| * + Before you can issue SQL commands to a database in the MySQL Browser, you need to designate which database you’re sending the commands to.
		- **USE** *databasename*;
		- Or, simply double-click the database in the MySQL Browser navigation pane.
 | USE alexamara |
| Creating a Table**CREATE TABLE** *tblTableName***(***fieldName datatype* [**NOT NULL**][**PRIMARY KEY**][**AUTO\_INCREMENT**],*fieldName datatype, fieldName datatype,repeat for all fields***);*** + SQL is a *free format* command language, which means you can insert extra spaces and carriage returns wherever you want. Take advantage of this to make your commands as readable as possible.
	+ SQL commands can be entered in any case (upper, lower, mixed)
	+ This command tells SQL to create a new table with the name you supplied.
	+ This command must also include the list of fields in the table, and the data types of those fields.
	+ The list of fields must be surrounded by (parentheses)
	+ Most implementations of SQL require a semicolon at the end of the command.
 |  |
| * + The **PRIMARY KEY** keyword designates this field as the primary key for the table.
		- This technique can only be used if there is one primary key field. See below to designate multi-field primary keys.
	+ The **NOT NULL** clause specifies that this field may not be left blank (NULL) or changed to a NULL value.
		- **PRIMARY** **KEY** fields are automatically designated **NOT** **NULL**.
 |  |
| * + Designating a Multi-Field Primary Key
		- List all the fields in the table, but don’t include the **PRIMARY** **KEY** designation.
		- After the last field, enter another comma, then enter the command **PRIMARY KEY**
		- Then, in (parentheses), list the field names included in the key.
		- **PRIMARY KEY** (*fieldName, fieldName, etc*)
 |  |
| * + The **AUTO\_INCREMENT** clause is optional.
		- This option can only be designated on **PRIMARY** **KEY** fields.
		- The data type is normally designated **INTEGER**
		- Values in this field are automatically incremented whenever new records are added to the table.
		- If records are deleted, their numbers are not reused
		- You must use **INSERT** [version 2](#insert2) to add records to tables with **AUTO\_INCREMENT** fields
 |  |
| * + SQL Data Types
		- **CHAR (***n***)**text data, *n* represents max charactersuse only if the field will always be full(fills with spaces otherwise)
		- **VARCHAR (***n***)**text data, *n* represents max characters but the number of characters can be less (no fill)
		- **TEXT** (~64KB)**MEDIUMTEXT** (~16MB)**LONGTEXT** (~4GB)MySQL equivalent of a memo
		- **DATETIME**date and time data
		- **DATE**date only (times provided are ignored)
		- **TIME**time only (dates provided are ignored)
		- **SMALLINT**whole number (±32K)
		- **INT or INTEGER**whole number (±2G)
		- **BIGINT** whole number (larger than 2G)
		- **DECIMAL(***p,q***) or DEC**number with decimal places*p* designates total number of digits (point not included)*q* designates the number of digits after decimal point
		- **BIT or BOOL or BOOLEAN**Use for Yes/No True/False fieldsEnter 1 or True; 0 or False when inserting data
 |  |
| * + Examples:**CREATE TABLE** cities**(** cityCode**INT****PRIMARY KEY AUTO\_INCREMENT***,* cityName **VARCHAR** (20) **NOT** **NULL**, population **INT**, numEmployees **SMALLINT**, taxRate **DECIMAL**(5,3), lastCensus **DATE,** polutionProblem **BOOLEAN** **);**
 | Create tables using [Alexamara Marina design](#sampledesign) |
| 1. Designating Foreign Keys
	* Often not necessary
	* Can help database ensure referential integrity (no child without parent).
	* Can also be used by some apps to reverse engineer an ERD
	* After the last field, enter another comma, then enter the command **FOREIGN KEY**
	* Then, in (parentheses), list the field name that links to another table (*foreign key*).
	* Add the keyword **REFERENCES**
	* List the table name that is linked to
	* After the table name, list the field in the parent table that corresponds to this field in the child table
	* Repeat for all foreign keys
	* **FOREIGN KEY** (fieldname) **REFERENCES** tblOther(fieldname)
 | FYI |
| 1. Automatically Setting Last Modified
	* For audit purposes, many databases store that last date and time a record was modified
		+ Some store who modified it
	* MySQL makes saving the last modified date easy lastModified TIMESTAMP DEFAULT now() ON UPDATE now();

 * + - This command defines a field, lastModified, with type

**TIMESTAMP** * + - When new records are created, the **DEFAULT** value of the field is set to the results of the function now() which returns the current date and time.
		- When a record is updated, lastModified is again set to the current date and time.

  | Information only |
| 1. Listing the Tables in the Database

**SHOW TABLES;*** + Lists all the tables in the database
	+ Tables are also shown in the Schemas pane (refresh if necessary)
	+ MySQL Browser converts all table names to lowercase to provide the best compatibility between different operating systems.
		- [Source](http://dev.mysql.com/doc/refman/5.6/en/identifier-case-sensitivity.html)
		- I don’t recommend you change this option
 | Try it. |
| 1. Showing a Table’s Structure
2. DESCRIBE *tblTableName*;
	* Shows each field’s name, data type, NULL acceptance, and primary key status.
	* Alternatively, you can use **SHOW COLUMNS FROM**
3. SHOW CREATE TABLE *tblTableName*;
	* Creates a one cell table that includes the entire command used to create the table
	* Great for exporting the table structure only
 | Try it. |
| 1. Removing a Table
2. DROP TABLE *tblTableName*;
	* Deletes a table and all the data in it from the database (note, there is no delete verification—be careful!)
	* Can also drop table from Schemas pane
 | Create table *junk* with a couple of junk fields.Drop junk table. |
| 1. Adding Data to a Table

**INSERT** **INTO** *tblTableName***VALUES****(**‘*chardata*’, *numdata, ‘2002-02-15’*, **NULL****);*** + This command adds a new record to the table specified.
	+ Repeat the command to enter multiple records
	+ **NOTE**: you must use [Version 2](#insert2) of the Insert command (see below) to insert records into tables with auto increment fields.
	+ Data must be listed in the order the fields were entered into the database (CREATE TABLE)
	+ String data must be surrounded by quotes (single or double) in MySQL
	+ Commas separate each data item (surrounding spaces not required)
	+ Dates are entered surrounded by quotes, in yyyy-mm-dd format
		- Alternatively (MySQL) you can enter the date in yyyymmdd format **(quotes optional)**
	+ Use hh:mm:ss format (military time) for **TIME** fields
		- Separate from date with a space in **DATETIME** fields
		- Seconds (ss) are optional
	+ Boolean values
		- 0 or false
		- 1 or true
	+ To leave a field blank, enter the keyword **NULL**
		- Don’t try this on **NOT NULL** fields. SQL will ignore the entire record.
	+ SQL recognizes duplicate primary keys and ignores any record with a duplicate key.
	+ Data entry errors can cause SQL to ignore your command.
 | Create a sample table with a couple of fields **without auto-increment** field.Insert some data.Drop. |
| 1. INSERT Command—Version 2
	* Sometimes, you have a partial record’s data available. Instead of entering NULL a bunch of times, you can tell SQL to only fill selected fields

**INSERT** **INTO** *tablename***(***fldName1, fldName2, fldName3,etc***)****VALUES****(***datafld1, datafld2, datafld3, etc***);*** + Be sure not to skip **NOT NULL** fields.
		- SQL actually enters the empty string into these fields (“”) instead of setting them to NULL
	+ Inserting into **AUTO\_INCREMENT** tables.
		- List all fields and values for those fields.
		- Leave the **AUTO\_INCREMENT** field off the field list; SQL will automatically fill in the next number in the sequence.
		- MySQL does allow you to provide a data value for an auto-increment field, but you should only use this if you’re importing data where linked records already exist.
 | Add sample data Marinas and MarinaSlips and ServiceRequestsAdd a partially filled record into tblServiceRequests |
| 1. Inserting Multiple Records
	* You can insert more than one record at a time with the **INSERT** statement.
	* After the first set of values, add a comma, then another set of values (in parentheses), and another and another.

**INSERT** **INTO** *tblTableName* **(***fldName1, fldName2, fldName3,etc***)****VALUES****(***datafld1, datafld2, datafld3, etc***),****(***datafld1, datafld2, datafld3, etc***),****(***datafld1, datafld2, datafld3, etc***),** **etc****(***datafld1, datafld2, datafld3, etc***);** | Add multiple records to tblMarinaSlips with one command |
| 1. Viewing Records

**SELECT \* FROM** *tblTableName***;*** + This command shows all fields in all records.
	+ We will discuss the SELECT command in greater detail later
 | Try it. |
| 1. Deleting Records

**DELETE FROM** *tblTableName***WHERE** *fieldName = criteriavalue***;*** + This command deletes a record (or records) from the specified table that matches the criteria.
	+ Warning, there is no way to undo this deletion and there is no delete verification.
	+ For this unit, I recommend you use the primary key in the WHERE clause
	+ Example:**DELETE FROM** cities**WHERE** cityId = 12**;**
	+ This command would delete the record where the cityId contains 12.
 | Enter a bogus record, then delete it. |
| 1. Correcting Data Errors
	* If there’s an error in one or more field values, you could delete the entire record or, you could simply change the incorrect value(s).

**UPDATE** *tblTableName***SET** *fieldName = newvalue***WHERE** *fieldName = criteriavalue***;*** + This command changes the value in a specified field, in a specified table, in the records that match the criteria, to a new value.
		- UPDATE *tblTableName* designates which table to make the changes to
		- SET *fieldName = newvalue* designates which field to change and what value to change it to (*newvalue*)
		- WHERE *fieldName = criteriavalue* designates which records to change: those where the specified field contains the appropriate value.
		- For this unit, I recommend you use the primary key in the WHERE clause
	+ Example:**UPDATE** cities**SET** cityName *=* ‘Stevens Point’**WHERE** cityId = 12**;**
		- This might be used to correct a typo (*Stevns Pointe)* in a record whose cityId field contains 12.
 | Modify the contents of a record |
| 1. Alexamara Sample Design

MarinasmarinaIdname (20)address (15)city (15)state (2)zip (5)marina slipsslipIdmarinaIdownerIdslipnum (4)lengthrentalfeeownersownerIdlastName (30)firstName(20)phone(10)serviceRequestsserviceIdslipIddescription(150)serviceDatecomplete (bool) |  |