

CSCI 2910 Client/Server-Side Programming

Topic: Intro to Database and SQL

Today's Goals

Today's lecture will cover:

- a basic introduction to databases
- a description of the client/server model and how it relates to databases
- an introduction to SQL and some of its commands
- instructions on how to log onto the Linux server and MySQL

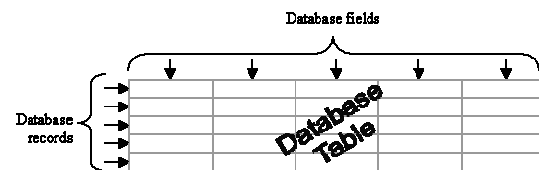
Basics of Databases

A database is made up of:

- fields – a compilation of types of information or properties
- records – a compilation of individual items with specific values for the aforementioned information or properties
- For example, a student record could contain fields such as student id, name, rank, street address, city, state, and zip code.
- A specific record might have a student id of 12345678, name of Jane Smith, rank of sophomore, street address of 9999 Buttermilk Lane, city of Johnson City, state of Tennessee, and a zip code of 37601.

Basics of Databases (continued)

- All of this information is contained in tables where the rows represent each record and the columns represent the fields.



"Hey, a table! That's kinda like a spreadsheet, right?"

- Unlike a spreadsheet, the rows (records) of a database must be independent of one another
- Unlike a spreadsheet, the columns (fields) of a database should be independent of one another
- Example: Gradebook with columns for each quiz, test, and homework grade.
 - Spreadsheet: one column might be used to compute the final grade
 - Database: Cannot have a field for this. Instead, just before you presented the data (results set), you would calculate a final grade to be presented. That value is never stored in a table.

In-Class Exercise

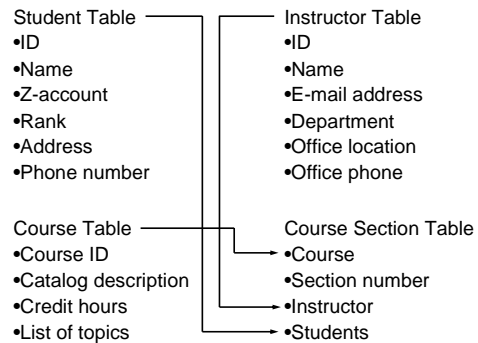
- In teams of 3 or 4, develop an idea for a table in a database that you would like to develop
- For this particular table, identify the fields along with some examples for records

Relational Databases

- A database may contain multiple tables too.
- For example, a database used for a section of a course may need to have a way to identify a student (student ID), but would not have to the student's personal information
- Therefore, the university's database would contain multiple tables:
 - Student information
 - Course information
 - Classroom information

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 7

Relational Databases (continued)



CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 8

Relational Databases (continued)

- Through proper interaction with the database, if an administrator wanted to get the z-accounts for all students taking CSCI 2910 section 001, he or she should be able to do it.
- There are a number of issues surrounding the proper design of a database – we will not be covering them in this class.
- The purpose of this introduction is to learn how to access or modify the information in an existing database.

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 9

Keys

- A key is a field by which records may be sorted.
- There are a number of uses for keys:
 - A primary key can be used to uniquely identify a record.
 - A common key is a key shared by two tables in a relational database.
 - A foreign key is a common key that can be used to identify records from another table.

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 10

Primary Keys

- Each record within a table must somehow be uniquely identifiable.
- For example, how can we make sure that we're looking at the correct student information in the student table?
- Answer: No two students share the same student id.
- Siblings may have the same parents, roommates may have the same address, but no one has identical student IDs.
- Therefore, we can use a field containing the student id to identify a specific record in the student database.
- This unique identification is called the *Primary Key*.

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 11

Simple Relational Database Example

Course Table

Department	Course	Section	Semester	Year	Instructor
CSCI	2800	001	Spring	2006	2
CSCI	2800	201	Spring	2006	1
CSCI	2910	001	Spring	2006	4
CSCI	2910	201	Spring	2006	3

Instructor Table

ID	Name	E-mail	Phone
1	Bailes	bailes@etsu.edu	423.439.6958
2	Bailey	baileyg@etsu.edu	423.439.6959
3	Laws	lawsm@etsu.edu	423.439.6952
4	Tarnoff	tarnoff@etsu.edu	423.439.6404

Primary keys

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 12

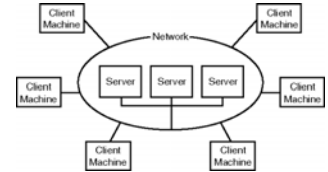
In-Class Exercise

- Using the same teams you had for the first exercise, identify the primary key for the table you developed earlier
- Create a second table that uses as one of its fields records from the first table.
- For this new table, identify the fields along with some examples for records

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 13

Client/Server Model

- Clients, typically PCs, provide an end user with access to the network.
- Servers are accessible from the network and provide services such as web pages or database access to the clients.



CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 14

Databases and the Client/Server Model

- Database systems typically reside on the server, but are not as part of the software providing server functionality.
- An interface must exist between server software and the database.
- Three tier architecture – Server/client model adds middle layer that handles transactions between client and database server.
- Middle layer provides:
 - ability to access more than one database with a single transaction
 - ability connect to many different types of data sources
 - ability to prioritize requests before they reach the data base
 - improved security

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 15

What is SQL?

(Adapted from material found at <http://www.atlasindia.com/sql.htm>)

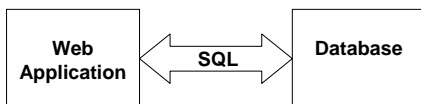
- Dr. Edgar F. Codd created a model for data storage that used a simple programming language to access the stored data
- In 1971, IBM used Dr. Codd's work to created a simple non-procedural language called Structured English Query Language (SEQUEL)
- In the late 80's, two standardization organizations (ANSI and ISO) developed a standardized version called Structured Query Language or SQL.

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 16

What is SQL? (continued)

(Adapted from material found at <http://www.atlasindia.com/sql.htm>)

- SQL is the language used to query all databases.
- It is a generic way to access the information in a database.
- Understanding SQL is vital to creating a database application such as a web interface.



CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 17

Different SQL Implementations

- There are multiple vendors of database products, each with their own implementation of SQL
- Each product should be compliant with ANSI standard
- Added features or commands do exists. These are called extensions.

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 18

Using SQL

- Assume that a database structure already exists, i.e., someone has already created tables for us containing fields and records.
- What sort of things might we want to do to this database?
 - Start/end a session with a specific database
 - Read a record
 - Insert a new record
 - Delete an existing record
 - Edit and restore an existing record

Querying Records

- A *query* is an inquiry to the database for information. This is done with SELECT.
- Syntax:

```
SELECT *| fieldname [, fieldnames]
FROM tablename [, tablenames] WHERE
fieldname=value ORDER BY fieldname [,
fieldnames]
```
- Example:

```
SELECT FirstName FROM Customers WHERE
LastName='Smith'
```

Data Manipulation

There are three basic commands to manipulate data:

- INSERT
- DELETE
- UPDATE

Adding a Record

- Syntax:

```
INSERT INTO tablename (fieldname [,
fieldnames]) VALUES (value [,
values])
```
- Example:

```
INSERT INTO Customers (FirstName,
LastName) VALUES ('Jane', 'Smith')
```

Removing a Record

- Syntax:

```
DELETE FROM tablename WHERE
fieldname=value
```
- Example:

```
DELETE FROM Customers WHERE
LastName='Jones'
```

Updating a Record

- Syntax:

```
UPDATE tablename SET
fieldname=value WHERE
fieldname=value
```
- Example:

```
UPDATE Customers SET
FirstName='Jeff' WHERE
LastName='Smith'
```

SQL Data Types

- The designer of a database can specify a data type for each field of a table.
- Different implementations of SQL support different types.
- There are four general types of data:
 - Numeric Types
 - Date and Time Types
 - String Types
 - Set Data Types

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 25

NULL Data Types

- In many cases, users need to have the option of leaving a field in a record blank. This is done by setting the field's value to NULL.
- NULL is the term used to represent a missing value. It is not the same as a 0 or a blank.
- NULL is also important when accessing or modifying data in a table.
- There are two methods for referencing a NULL value:
 - NULL (the keyword NULL itself)
 - '' (single quotation marks with nothing in between)

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 26

MySQL Numeric Data Types

(from <http://dev.mysql.com/doc/refman/5.0/en/numeric-type-overview.html>)

- **BIT[(M)]** – A bit-field type. *M* indicates the number of bits per value, from 1 to 64. The default is 1 if *M* is omitted.
- **TINYINT[(M)] [UNSIGNED] [ZEROFILL]** – A very small integer. The signed range is -128 to 127. The unsigned range is 0 to 255.
- **BOOL, BOOLEAN** – These types are synonyms for TINYINT(1). A value of zero is considered false. Non-zero values are considered true.
- **SMALLINT[(M)] [UNSIGNED] [ZEROFILL]** – A small integer. The signed range is -32768 to 32767. The unsigned range is 0 to 65535.
- **MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]** – A medium-sized integer. The signed range is -8388608 to 8388607. The unsigned range is 0 to 16777215.
- **INTEGER[(M)], INT[(M)] [UNSIGNED] [ZEROFILL]** – A normal-size integer. The signed range is -2147483648 to 2147483647. The unsigned range is 0 to 4294967295.
- **BIGINT[(M)] [UNSIGNED] [ZEROFILL]** – A large integer. The signed range is -9223372036854775808 to 9223372036854775807. The unsigned range is 0 to 18446744073709551615.

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 27

More MySQL Numeric Data Types

(from <http://dev.mysql.com/doc/refman/5.0/en/numeric-type-overview.html>)

- **FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]** – A small (single-precision) floating-point number. Allowable values are -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38. *M* is the total number of decimal digits and *D* is the number of digits following the decimal point.
- **DOUBLE[(M,D)], DOUBLE PRECISION[(M,D)], or REAL[(M,D)] [UNSIGNED] [ZEROFILL]** – A normal-size (double-precision) floating-point number. Allowable values are -1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308. *M* is the total number of decimal digits and *D* is the number of digits following the decimal point.
- **FLOAT(p) [UNSIGNED] [ZEROFILL]** – A floating-point number. *p* represents the precision in bits, but MySQL uses this value only to determine whether to use FLOAT or DOUBLE for the resulting data type.
- **DEC[(M,D)], DECIMAL[(M,D)], NUMERIC[(M,D)], or FIXED[(M,D)] [UNSIGNED] [ZEROFILL]** – A packed "exact" fixed-point number. *M* is the total number of decimal digits (the precision) and *D* is the number of digits after the decimal point (the scale).

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 28

MySQL Date & Time Data Types

(<http://dev.mysql.com/doc/refman/5.0/en/date-and-time-type-overview.html>)

- **DATE** – A date. The supported range is '1000-01-01' to '9999-12-31'.
- **DATETIME** – A date and time combination. The supported range is '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
- **TIMESTAMP[(M)]** – A timestamp. The range is '1970-01-01 00:00:00' to partway through the year 2037. A **TIMESTAMP** column is useful for recording the date and time of an **INSERT** or **UPDATE** operation.
- **TIME** – A time. The range is '-838:59:59' to '838:59:59'.
- **YEAR[(2|4)]** – A year in two-digit or four-digit format. The default is four-digit format. In four-digit format, the allowable values are 1901 to 2155, and 0000. In two-digit format, the allowable values are 70 to 69, representing years from 1970 to 2069.

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 29

MySQL String Data Types

(<http://dev.mysql.com/doc/refman/5.0/en/string-type-overview.html>)

- **CHAR(M) [BINARY | ASCII | UNICODE]** – A fixed-length string that is always right-padded with spaces to the specified length when stored. *M* represents the column length. If *M* isn't specified, default is 1.
- **VARCHAR(M) [BINARY]** – A variable-length string. *M* represents the maximum column length.
- **TEXT[(M)]** – A **TEXT** column with a maximum length of 65,535 ($2^{16} - 1$) characters.
- **MEDIUMTEXT** – A **TEXT** column with a maximum length of 16,777,215 ($2^{24} - 1$) characters.
- **LONGTEXT** – A **TEXT** column with a maximum length of 4,294,967,295 or 4GB ($2^{32} - 1$) characters. The maximum length is limited by maximum packet size of protocol used.

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 30

MySQL Set Data Types

(<http://dev.mysql.com/doc/refman/5.0/en/string-type-overview.html>)

- ENUM('value1','value2',...) – An enumeration. A string object that can have only one value, chosen from the list of values 'value1', 'value2', ..., NULL or the special '' error value. An ENUM column can have a maximum of 65,535 distinct values. ENUM values are represented internally as 16-bit integers.
- SET('value1','value2',...) – A set. A string object that can have zero or more values, each of which must be chosen from the list of values 'value1', 'value2', ... A SET column can have a maximum of 64 members. SET values are represented internally as 64-bit integers.

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 31

In-Class Exercise

- As a class, suggest which data types would be best suited for some of the proposed fields from earlier exercises.

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 32

Conducting an SQL Session

- There are many different ways to conduct an SQL session
- Basically, the user needs to access the server, then connect to a specific database
- This can be done either through a special syntax in the server-side application or through command line commands

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 33

Connecting to the Databases

- For our command line SQL work, we will be using the CSCI server einstein.etsu.edu
- Einstein is a Linux server. You should have been given an account when you registered for the class.
- Getting access to MySQL is a two step process:
 - First, log onto the linux box
 - Second, log onto the MySQL server

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 34

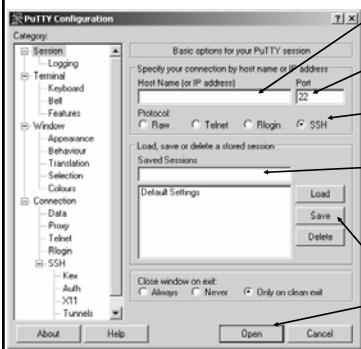
Logging onto the Linux Box

- All of the laboratory machines should have putty.exe installed. You'll find it under "Internet Tools."
- Opening Putty should present you with a window like that shown on the following slide.

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 35

Using Putty



Step 1: Enter "einstein.etsu.edu" under Host Name.

Step 2: Make sure the port selected is 22.

Step 3: Make sure the protocol selected is SSH.

Step 4: Enter the name "Einstein" in the Saved Sessions field. (This will help us identify it later.)

Step 5: Press the button labeled "Save".

Step 6: Press the button labeled "Open". This will begin your session.

CSCI 2910 – Client/Server-Side Programming

Intro to DB and SQL – Page 36

Logging onto Einstein

- If you've successfully used Putty to open a connection to Einstein, you should see a text window with a prompt like "login as:" at the top of the window.
- At the prompt, enter your user name (z-name), then press Enter.
- You will then be prompted for your password. This password should have been sent to you toward the beginning of the semester when Robert Nielsen set up your accounts.
- Pressing Enter should log you onto Einstein.

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 37

Logging onto MySQL

- Once logged onto Einstein, you should have a prompt that looks like:

```
[zabc123@einstein ~]$
```

- At this prompt, type:

```
mysql -u zabc123 -p
```

Application to run Switch to indicate user name follows User name Tells MySQL to prompt for password

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 38

Passwords & Logging Off

- To change your Einstein password, type "passwd" at the **Einstein** prompt and follow the directions.
- To change your MySQL password, type the following command at the **MySQL** prompt inserting your new password for "new_pw".
`SET PASSWORD = PASSWORD('new_pw');`
- To log out of MySQL, type "exit" and press Enter.
- To log out of Einstein, type "logout" and press Enter.

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 39

Assignment

- By next Tuesday, make sure you are able to log onto Einstein and subsequently log onto MySQL.
- Remember that there will be a quiz on Tuesday covering the material in this lecture.

CSCI 2910 – Client/Server-Side Programming Intro to DB and SQL – Page 40