CSCI 2910
Client/Server-Side Programming

Topic: More on SQL

Reading: *PHP and MySQL*, pp. 152–168

Today’s Goals

- Today’s lecture will improve our use of the SQL query "SELECT" by using:
  - WHERE,
  - BETWEEN,
  - IN,
  - LIKE,
  - NOT,
  - arithmetic operators, and
  - LIMIT.
- The concept of joining tables will also be introduced.

More on Querying Records

- Remember that the syntax for the select command is:

```
SELECT [ ALL | DISTINCT] *| COLUMN1[,
COLUMN2 ] FROM TABLE1 [ , TABLE2 ] WHERE
[ CONDITION1 | EXPRESSION1 ][ AND|OR
CONDITION2 | EXPRESSION2 ] ORDER BY
fieldname [, fieldnames] [ASC|DESC]
```

- * allows us to view all fields
- FROM identifies the table in which we're interested
- WHERE allows us to restrict the records we're looking at.
- ORDER BY allows us to sort the output

Conditions

- In the WHERE keyword of the SELECT command, a condition is required to limit the returned records.
- The condition evaluates to TRUE or FALSE for each record.
- Records with a value of TRUE for the condition are retrieved from the query.
- There can be more than one condition in the WHERE clause, connected by the AND and OR operators.

Conditions (continued)

- An operator is a character or keyword in SQL that is used to combine elements in a SQL statement.
- Examples:

```
SELECT * FROM students WHERE ID =
  10194356;
SELECT LASTNAME FROM students WHERE AGE
  > 24;
```

Conditions (continued)

SQL has six relational operators that can be used to create conditions:

=	Equal
!=	Not equal
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Conditions (continued)

- For example, if we executed the following SELECT command on the mylibrary table that we created during lab, we would pull all books published before 1965.

```
SELECT * FROM mylibrary WHERE PUB_YEAR < 1965;
```

```
+-------------------------------+---------------+--------+------+---------+--------+
| TITLE                         | AUTHOR        |PUB_YEAR| PRICE| INDX    |ON_SHELF|
+-------------------------------+---------------+--------+------+---------+--------+
| Catcher in the Rye, The       | J.D. Salinger |   1951 | 6.99 | F-SAL00 |      1 |
| One Flew Over the Cuckoos Nest| Ken Kesey     |   1963 | 7.99 | F-KES00 |      1 |
| Fahrenheit 451                | Ray Bradbury  |   1953 | 6.99 | F-BRA00 |      1 |
+-------------------------------+---------------+--------+------+---------+--------+
```

Conditions (continued)

- Conditions may also be combined with AND and OR.
- For example, if we executed the following SELECT command on the mylibrary table that we created during lab, we would pull all books published before 1965 that are also less than $7.00.

```
SELECT * FROM mylibrary WHERE (PUB_YEAR < 1965
AND PRICE < 7);
```

```
+-------------------------+---------------+----------+-------+---------+--------+
| TITLE                   | AUTHOR        | PUB_YEAR | PRICE | INDX    |ON_SHELF|
+-------------------------+---------------+----------+-------+---------+--------+
| Catcher in the Rye, The | J.D. Salinger |     1951 |  6.99 | F-SAL00 |      1 |
| Fahrenheit 451          | Ray Bradbury  |     1953 |  6.99 | F-BRA00 |      1 |
+-------------------------+---------------+----------+-------+---------+--------+
```

Identifying NULL Values

- To identify records with NULL in their fields, the condition "IS NULL" must be used, not "=NULL"
- Example: Assume we inserted a record with a NULL value for the price. The following shows the results of using "IS NULL" versus "=NULL".

```
mysql> SELECT * FROM mylibrary WHERE PRICE IS NULL;
+---------------+---------------+----------+-------+---------+----------+
| TITLE         | AUTHOR        | PUB_YEAR | PRICE | INDX    | ON_SHELF |
+---------------+---------------+----------+-------+---------+----------+
| PHP and MySQL | Hugh Williams |     2004 |  NULL | T-WIL00 |        1 |
+---------------+---------------+----------+-------+---------+----------+
1 row in set (0.00 sec)

mysql> SELECT * FROM mylibrary WHERE PRICE = NULL;
Empty set (0.00 sec)
```

BETWEEN

- The BETWEEN operator is used to search for values that are between given minimum and maximum values.
- Syntax: WHERE *fieldname* BETWEEN *min* AND *max*
- Example:

```
mysql> SELECT * FROM mylibrary WHERE PRICE BETWEEN 6 AND 7;
+-------------------------+----------------+----------+-------+---------+--------+
| TITLE                   | AUTHOR         | PUB_YEAR | PRICE | INDX    |ON_SHELF|
+-------------------------+----------------+----------+-------+---------+--------+
| Catcher in the Rye, The | J.D. Salinger  |     1951 |  6.99 | F-SAL00 |      1 |
| Fahrenheit 451          | Ray Bradbury   |     1953 |  6.99 | F-BRA00 |      1 |
| Carrie                  | Stephen King   |     1974 |  6.95 | F-KIN00 |      1 |
| Jaws                    | Peter Benchley |     1974 |  6.99 | F-BEN00 |      1 |
| 158-Pound Marriage, The | John Irving    |     1973 |  6.99 | F-IRV00 |      1 |
+-------------------------+----------------+----------+-------+---------+--------+
5 rows in set (0.00 sec)
```

IN

- The IN operator is used to compare a value to a list of literal values that have been specified.
- A TRUE IS returned when the compared value is contained in the list.
- Example:

```
mysql> SELECT * FROM mylibrary WHERE PUB_YEAR IN (1953, 1974, 1965);
+----------------+----------------+----------+-------+---------+--------+
| TITLE          | AUTHOR         | PUB_YEAR | PRICE | INDX    |ON_SHELF|
+----------------+----------------+----------+-------+---------+--------+
| Fahrenheit 451 | Ray Bradbury   |     1953 |  6.99 | F-BRA00 |      1 |
| Carrie         | Stephen King   |     1974 |  6.95 | F-KIN00 |      1 |
| Jaws           | Peter Benchley |     1974 |  6.99 | F-BEN00 |      1 |
| Hobbit, The    | J.R.R. Tolkien |     1965 |  7.99 | F-TOL00 |      1 |
+----------------+----------------+----------+-------+---------+--------+
4 rows in set (0.02 sec)
```

LIKE and Wildcards

- The LIKE operator is used in conjunction with wildcard operators to identify values satisfying less restrictive conditions, e.g., all student ids beginning with "1012".
- Wildcard operators:
  - % represents 0, 1, 2, or more digits or characters
  - _ (underscore) represents exactly one digit or character
- Wildcards can be used in combinations to search for specific patterns
- Examples:
  - WHERE LAST_NAME LIKE T% identifies last names beginning with 'T'
  - WHERE FIRST_NAME LIKE _ILL% identifies first names where the 2nd, 3rd, and 4th letters are "ILL", e.g., Will, Bill, Willy, William, Billy, Gill, etc.
  - WHERE STUDENT_ID LIKE %55% identifies student ids that contain the string "55"
  - WHERE STUDENT_ID LIKE _01% identifies student ids where the second and third digit are 0 and 1.

LIKE and Wildcards (continued)

For example, if we executed the following SELECT command on the mylibrary table, we would pull all books published in the 70's.

```
SELECT * FROM mylibrary
WHERE PUB_YEAR LIKE '197_';
```

```
+-----------------------------+---------------+--------+-------+---------+--------+
| TITLE                       | AUTHOR        |PUB_YEAR| PRICE | INDX    |ON_SHELF|
+-----------------------------+---------------+--------+-------+---------+--------+
| Carrie                      | Stephen King  |   1974 |  6.95 | F-KIN00 |      1 |
| Jaws                        | Peter Benchley|   1974 |  6.99 | F-BEN00 |      1 |
| 158-Pound Marriage, The     | John Irving   |   1973 |  6.99 | F-IRV00 |      1 |
| World According to Garp, The| John Irving   |   1978 |  7.99 | F-IRV01 |      1 |
+-----------------------------+---------------+--------+-------+---------+--------+
```

LIKE and Wildcards (continued)

For example, if we executed the following SELECT command on the mylibrary table, we would pull all books with an index starting with 'T'.

```
SELECT * FROM mylibrary WHERE INDX LIKE 'T%';
```

```
+---------------------------+------------------+--------+--------+--------+--------+
| TITLE                     | AUTHOR           |PUB_YEAR| PRICE  | INDX   |ON_SHELF|
+---------------------------+------------------+--------+--------+--------+--------+
| Road Ahead, The           | Bill Gates       |   1996 |  14.99 | T-GAT00|      1 |
| Computer Organization     | Carl Hamacher    |   2001 | 132.81 | T-HAM00|      1 |
| Linux Bible, 2005 Edition | Christopher Negus|   2005 |  26.39 | T-NEG00|      1 |
| Linux For Dummies, 6th Ed.| Dee-Ann LeBlanc  |   2005 |  20.79 | T-LEB00       1 |
+---------------------------+------------------+--------+--------+--------+---_----+
```

Escape Characters

- If values to be examined in the database include wildcard characters, they can be "escaped" with a backslash.
- For example:

```
SELECT * FROM mylibrary
WHERE INDX LIKE '\_%';
```

retrieves records from mylibrary with an INDX starting with an underscore.

NOT

- The NOT operator reverses the meaning of the logical operator with which it is used.
- The NOT can be used with the following operators in the following methods:
  - NOT EQUAL
  - NOT BETWEEN
  - NOT IN
  - NOT LIKE
  - IS NOT NULL

Arithmetic Operators

- Arithmetic operators are used to perform mathematical functions on values in SQL.
- There are four conventional operators for mathematical functions.
  - + (addition)
  - - (subtraction)
  - * (multiplication)
  - / (division)

Arithmetic Operators (continued)

Assume we have a database of products with a table as shown below:

```
SELECT * FROM products;
+---------+------------------------+---------+-----------+----------+
| PROD_ID | PROD_NAME              | RETAIL  | WHOLESALE | SHIPPING |
+---------+------------------------+---------+-----------+----------+
| G132A   | Cannon Digital Camera  | 1499.99 |    899.99 |    12.00 |
| D816D   | Epson LCD Projector    | 1699.99 |   1199.99 |    13.50 |
| H724G   | Sony LCD TV            | 1549.99 |   1399.99 |    30.00 |
| K632H   | Apple 30 GB iPod       |  279.99 |    199.99 |     6.50 |
| I543J   | Archos Multimedia Plyr |  649.99 |    599.99 |     6.50 |
| T556Y   | Palm TX Handheld       |  265.99 |    239.99 |     6.50 |
| E663E   | Sony Mini DV Handicam  |  349.99 |    299.99 |    18.50 |
| V875C   | Apple MacBook Pro      | 2499.99 |   1999.99 |    18.00 |
| H083V   | Epson Photo Scanner    |  419.99 |    375.99 |    16.50 |
| U996D   | MS XP Pro w/SP 2       |  189.99 |    159.99 |     9.50 |
+---------+------------------------+---------+-----------+----------+
```

Arithmetic Operators (continued)

To display the retail cost of each product with shipping, use the '+' to combine RETAIL with SHIPPING.

```
mysql> SELECT PROD_NAME, RETAIL+SHIPPING FROM products;
+------------------------+-----------------+
| PROD_NAME              | RETAIL+SHIPPING |
+------------------------+-----------------+
| Cannon Digital Camera  |         1511.99 |
| Epson LCD Projector    |         1713.49 |
| Sony LCD TV            |         1579.99 |
| Apple 30 GB iPod       |          286.49 |
| Archos Multimedia Plyr |          656.49 |
| Palm TX Handheld       |          272.49 |
| Sony Mini DV Handicam  |          368.49 |
| Apple MacBook Pro      |         2517.99 |
| Epson Photo Scanner    |          436.49 |
| MS XP Pro w/SP 2       |          199.49 |
+------------------------+-----------------+
```

Arithmetic Operators (continued)

To display the profit, use the '–' to subtract the WHOLESALE cost from the RETAIL price.

```
mysql> SELECT PROD_NAME, RETAIL-WHOLESALE FROM products;
+------------------------+------------------+
| PROD_NAME              | RETAIL-WHOLESALE |
+------------------------+------------------+
| Cannon Digital Camera  |           600.00 |
| Epson LCD Projector    |           500.00 |
| Sony LCD TV            |           150.00 |
| Apple 30 GB iPod       |            80.00 |
| Archos Multimedia Plyr |            50.00 |
| Palm TX Handheld       |            26.00 |
| Sony Mini DV Handicam  |            50.00 |
| Apple MacBook Pro      |           500.00 |
| Epson Photo Scanner    |            44.00 |
| MS XP Pro w/SP 2       |            30.00 |
+------------------------+------------------+
```

Arithmetic Operators (continued)

Constants can also be used with arithmetic operators. The query below multiplies RETAIL by 80%.

```
mysql> SELECT PROD_NAME, RETAIL*0.8 FROM products;
+------------------------+------------+
| PROD_NAME              | RETAIL*0.8 |
+------------------------+------------+
| Cannon Digital Camera  |    1199.99 |
| Epson LCD Projector    |    1359.99 |
| Sony LCD TV            |    1239.99 |
| Apple 30 GB iPod       |     223.99 |
| Archos Multimedia Plyr |     519.99 |
| Palm TX Handheld       |     212.79 |
| Sony Mini DV Handicam  |     279.99 |
| Apple MacBook Pro      |    1999.99 |
| Epson Photo Scanner    |     335.99 |
| MS XP Pro w/SP 2       |     151.99 |
+------------------------+------------+
```

LIMIT

- The LIMIT operator is used to specify a subset of the output from a query.
- Syntax: LIMIT [*start,*] *size*
- Example:

```
mysql> SELECT * FROM products LIMIT 3;
+---------+-----------------------+---------+-----------+----------+
| PROD_ID | PROD_NAME             | RETAIL  | WHOLESALE | SHIPPING |
+---------+-----------------------+---------+-----------+----------+
| G132A   | Cannon Digital Camera | 1499.99 |    899.99 |    12.00 |
| D816D   | Epson LCD Projector   | 1699.99 |   1199.99 |    13.50 |
| H724G   | Sony LCD TV           | 1549.99 |   1399.99 |    30.00 |
+---------+-----------------------+---------+-----------+----------+
```

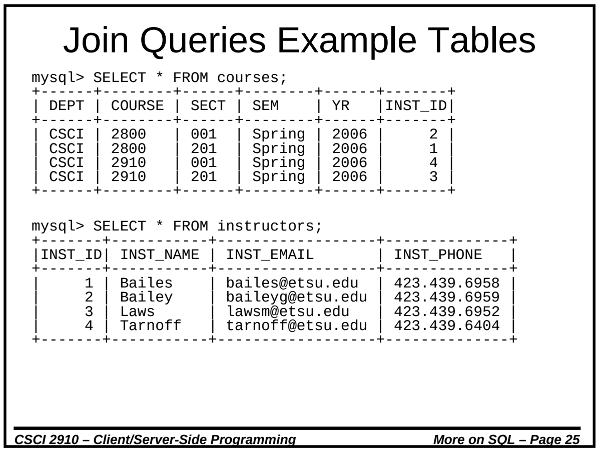
LIMIT (continued)

- By giving a starting index, the subset can be pulled from a specific location within the table.
- The index of the first record is 0.
- Example:

```
mysql> SELECT * FROM products LIMIT 2,3;
+---------+------------------------+---------+-----------+----------+
| PROD_ID | PROD_NAME              | RETAIL  | WHOLESALE | SHIPPING |
+---------+------------------------+---------+-----------+----------+
| H724G   | Sony LCD TV            | 1549.99 |   1399.99 |    30.00 |
| K632H   | Apple 30 GB iPod       |  279.99 |    199.99 |     6.50 |
| I543J   | Archos Multimedia Plyr |  649.99 |    599.99 |     6.50 |
+---------+------------------------+---------+-----------+----------+
```

Join Queries

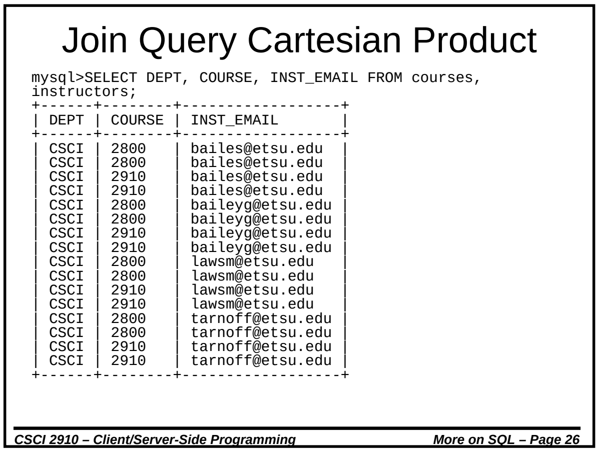
- The idea behind relational databases is that the tables have some field that provides a relation between records.
- The tables being joined are listed after the FROM clause.
- Several operators can be used to join tables such as =, <, >, <>, <=, >=,!=, BETWEEN, LIKE, and NOT
- The most common operator is the equal symbol.
- NOTE: If no "WHERE" condition is used, the Cartesian Product of the two tables will be returned.



Join Queries Example Tables

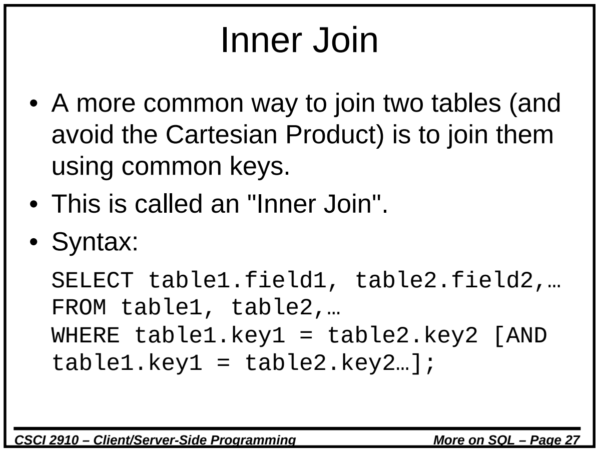
```
mysql> SELECT * FROM courses;
+------+--------+------+--------+------+-------+
| DEPT | COURSE | SECT | SEM    | YR   |INST_ID|
+------+--------+------+--------+------+-------+
| CSCI | 2800   | 001  | Spring | 2006 |     2 |
| CSCI | 2800   | 201  | Spring | 2006 |     1 |
| CSCI | 2910   | 001  | Spring | 2006 |     4 |
| CSCI | 2910   | 201  | Spring | 2006 |     3 |
+------+--------+------+--------+------+-------+

mysql> SELECT * FROM instructors;
+-------+-----------+------------------+--------------+
|INST_ID| INST_NAME | INST_EMAIL       | INST_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 |
+-------+-----------+------------------+--------------+
```



Join Query Cartesian Product

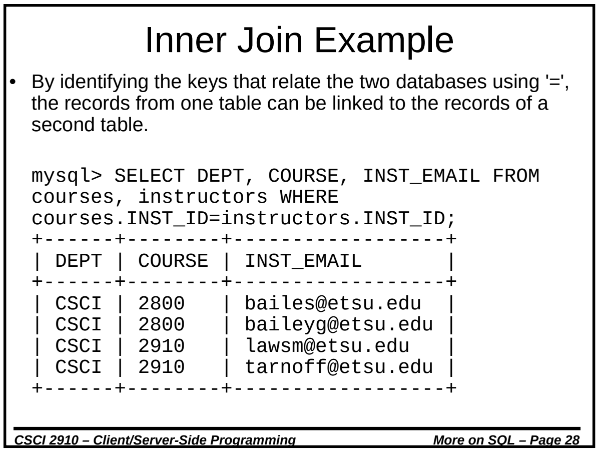
```
mysql>SELECT DEPT, COURSE, INST_EMAIL FROM courses,
instructors;
+------+--------+------------------+
| DEPT | COURSE | INST_EMAIL       |
+------+--------+------------------+
| CSCI | 2800   | bailes@etsu.edu  |
| CSCI | 2800   | bailes@etsu.edu  |
| CSCI | 2910   | bailes@etsu.edu  |
| CSCI | 2910   | bailes@etsu.edu  |
| CSCI | 2800   | baileyg@etsu.edu |
| CSCI | 2800   | baileyg@etsu.edu |
| CSCI | 2910   | baileyg@etsu.edu |
| CSCI | 2910   | baileyg@etsu.edu |
| CSCI | 2800   | lawsm@etsu.edu   |
| CSCI | 2800   | lawsm@etsu.edu   |
| CSCI | 2910   | lawsm@etsu.edu   |
| CSCI | 2910   | lawsm@etsu.edu   |
| CSCI | 2800   | tarnoff@etsu.edu |
| CSCI | 2800   | tarnoff@etsu.edu |
| CSCI | 2910   | tarnoff@etsu.edu |
| CSCI | 2910   | tarnoff@etsu.edu |
+------+--------+------------------+
```



Inner Join

- A more common way to join two tables (and avoid the Cartesian Product) is to join them using common keys.
- This is called an "Inner Join".
- Syntax:

```
SELECT table1.field1, table2.field2,…
FROM table1, table2,…
WHERE table1.key1 = table2.key2 [AND
table1.key1 = table2.key2…];
```



Inner Join Example

- By identifying the keys that relate the two databases using '=', the records from one table can be linked to the records of a second table.

```
mysql> SELECT DEPT, COURSE, INST_EMAIL FROM
courses, instructors WHERE
courses.INST_ID=instructors.INST_ID;
+------+--------+------------------+
| DEPT | COURSE | INST_EMAIL       |
+------+--------+------------------+
| CSCI | 2800   | bailes@etsu.edu  |
| CSCI | 2800   | baileyg@etsu.edu |
| CSCI | 2910   | lawsm@etsu.edu   |
| CSCI | 2910   | tarnoff@etsu.edu |
+------+--------+------------------+
```