SQL

CSE 190 M (Web Programming) Spring 2007
University of Washington

Reading: Sebesta Ch. 14 sections 14.1 - 14.2, 14.4, 14.6
References: SQL syntax reference, w3schools tutorial

Relational databases

- relational database: A method of structuring data as tables associated to each other by shared attributes.
- a table row corresponds to a record (tuple); a column corresponds to an attribute (field) of the record
- relational databases typically use Structured Query Language (SQL) to define, manage, and search data

Why databases?

- powerful: can search it quickly, filter data, combine data from multiple sources
- big: scale well up to very large data sizes
- safe: built-in mechanisms for failure recovery (transactions)
- multi-user: concurrency features let many users view/edit data at same time
- abstract: provides layer of abstraction between stored data and app(s)
    - many database programs understand the same SQL commands

Database software

- Oracle database
- Microsoft SQL Server (powerful) and Microsoft Access (simple)
- IBM DB2
- PostgreSQL (powerful/complex free open-source database system)
- MySQL (simple free open-source database system)
    - many "LAMP" servers run Linux, Apache, MySQL, and PHP
    - Wikipedia is run on PHP and MySQL
    - we will use MySQL in this course

Database design

- database design : the act of deciding the schema for a database
- database schema: a description of what tables a database should have, what columns each table should contain, which columns' values must be unique, etc.
- some database design principles:
  - keep it simple, stupid
  - eliminate redundancy, especially redundancy of lengthy data (strings)

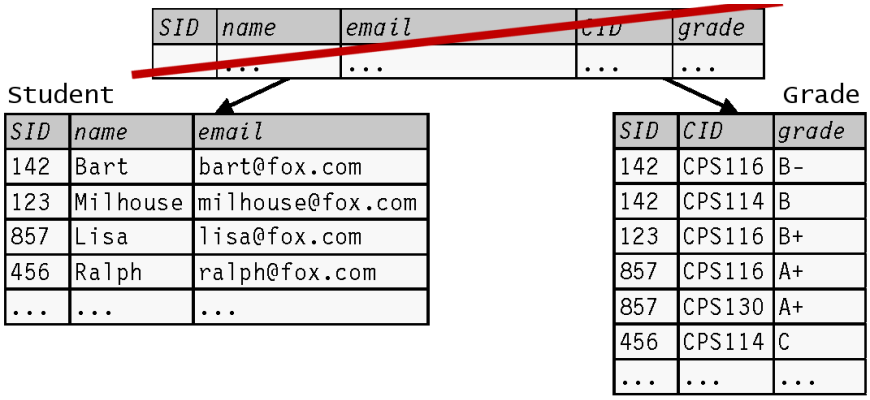
First database design

StudentGrade

SID	name	email	CID	grade
142	Bart	bart@fox.com	CPS116	B-
142	Bart	bart@fox.com	CPS114	B
123	Milhouse	milhouse@fox.com	CPS116	B+
857	Lisa	lisa@fox.com	CPS116	A+
857	Lisa	lisa@fox.com	CPS130	A+
456	Ralph	ralph@fox.com	CPS114	C
...	...	...	...	...

- what's good and bad about this design?
- uses only one table, but contains redundancy

Second database design



- splitting data into two tables (linked by Student ID) avoids redundancy
- this is also called normalizing the database
- normalized tables are often linked by unique integer IDs

Structured Query Language (SQL)

```
SELECT name FROM Student WHERE SID = 456;
INSERT INTO Grade VALUES ('123', 'CPS130', 'C+');
```

- a language for searching and updating a database
- a standard syntax that is used by all database software (with minor incompatiblities)
- a declarative language: describes what data you are seeking, not exactly how to find it

The SQL SELECT statement

```
SELECT column(s) FROM table;
```

```
SELECT SID, CID FROM Grade;
```

SID	CID
142	CPS116
142	CPS114
123	CPS116
857	CPS116
857	CPS130
456	CPS114

- the SELECT statement searches a database and returns a set of results
  - the column name(s) written after SELECT filter which parts of the rows are returned
  - table and column names are case-sensitive
  - * keeps all columns

Issuing SQL commands directly in MySQL

```
% mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.

mysql> USE Simpsons;
Database changed

mysql> SELECT * FROM Student;
+-----+----------+------------------+
| SID | name     | email            |
+-----+----------+------------------+
| 123 | Milhouse | milhouse@fox.com |
| 142 | Bart     | bart@fox.com     |
| 456 | Ralph    | ralph@fox.com    |
| 857 | Lisa     | lisa@fox.com     |
+-----+----------+------------------+
```

- other commands:
  - `SHOW TABLES;`
  - `SHOW DATABASES;`

The `DISTINCT` modifier

```
SELECT DISTINCT column(s) FROM table;
```

```
SELECT SID FROM Grade;
```

SID
142
142
123
857
857
456

```
SELECT DISTINCT SID FROM Grade;
```

SID
142
123
857
456

- eliminates duplicates from the result set

The WHERE clause

```
SELECT column(s) FROM table WHERE condition(s);
```

```
SELECT CID, grade FROM Grade WHERE SID = 142;
```

CID	grade
CPS116	B-
CPS114	B

- WHERE clause filters out rows based on their columns' data values
- in large databases, it's critical to use a WHERE clause to reduce the result set size
- suggestion: when trying to write a query, think of the FROM part first, then the WHERE part, and lastly the SELECT part

More about the WHERE clause

```
WHERE column operator value(s)
```

```
SELECT * FROM Grade WHERE grade <> 'A+';
```

SID	CID	grade
142	CPS116	B-
142	CPS114	B
123	CPS116	B+
456	CPS114	C

- the WHERE portion of a SELECT statement can use the following operators:
  - =, >, >=, <, <=
  - <> : not equal
  - BETWEEN min AND max
  - LIKE pattern
  - IN (value, value, ..., value)

Multiple `WHERE` clauses: AND, OR

```
SELECT * FROM Grade WHERE grade <> 'A+' AND SID <= 142;
```

SID	CID	grade
142	CPS116	B-
142	CPS114	B
456	CPS114	C

- multiple `WHERE` conditions can be combined using `AND` and `OR`

Approximate matches: `LIKE`

```
WHERE column LIKE pattern
```

```
SELECT * FROM Grade WHERE grade LIKE 'B%';
```

SID	CID	grade
142	CPS116	B-
142	CPS114	B
123	CPS116	B+

- `LIKE 'text%'` searches for text that starts with a given prefix
- `LIKE '%text'` searches for text that ends with a given suffix
- `LIKE '%text%'` searches for text that contains a given substring

Sorting: ORDER BY

```
ORDER BY column(s)
```

```
SELECT * FROM Grade WHERE grade LIKE 'B%' ORDER BY CID;
```

SID	CID	grade
142	CPS114	B
142	CPS116	B-
123	CPS116	B+

- sorts the result set by a given column
- can write ASC or DESC to sort in ascending (default) or descending order:

```
SELECT * FROM Grade ORDER BY CID DESC;
```

- can specify multiple orderings in decreasing order of significance:

```
SELECT * FROM Grade ORDER BY CID DESC, SID;
```

Connecting to MySQL in PHP: mysql_connect

```
$db = mysql_connect("host", "username", "password");
mysql_select_db("database name");

# connect to Simpsons database on local computer
$db = mysql_connect("localhost", "stepp", "6uldv8");
mysql_select_db("simpsons");
```

- mysql_connect opens connection to database on its server
  - any/all of the 3 parameters can be omitted (default: localhost, anonymous)
- mysql_select_db sets which database to examine

Error-checking: `mysql_error`

```
# connect to Simpsons database on local computer
$db = mysql_connect("localhost", "stepp", "6uldv8");
if (!$db) {
    die("A SQL error occurred: " . mysql_error());
}
if (!mysql_select_db("simpsons")) {
    die("A SQL error occurred: " . mysql_error());
}
```

- SQL commands can fail for a variety of reasons
  - database not reachable, wrong username/password, bad query syntax, ...
- for debugging, always test the results of PHP's `mysql` functions
  - if they are `FALSE` (`NULL`), print `mysql_error` result to see what failed

Reading result data: `mysql_query`

```
$db = mysql_connect("host", "username", "password");
mysql_select_db("database name");
$results = mysql_query("SQL query");
while ($row = mysql_fetch_array($results)) {
    do something with $row;
}
```

- `mysql_query` sends SQL query to database and returns results
- `mysql_fetch_array` returns one result row as an associative array
  - the column names are its keys, and the record's values are its values

PHP MySQL example

```
# connect to Simpsons database on local computer
$db = mysql_connect("localhost", "stepp", "6uldv8");
mysql_select_db("Simpsons");
$results = mysql_query("SELECT * FROM Grade WHERE SID = 142;");

# loop through each of Bart's course grade records
while ($row = mysql_fetch_array($results)) {
    print("Course ID: {$row['cid']}\n");
}
```

Checking rows returned: mysql_num_rows

```
# connect to Simpsons database on local computer
$db = mysql_connect("localhost", "stepp", "6uldv8");
mysql_select_db("Simpsons");
$results = mysql_query("SELECT * FROM Grade WHERE SID = 142;");

# check whether Bart took any courses
if (mysql_num_rows($results) == 0) {
    print("Student 142 did not take any classes.\n");
    ...
}
```

- `mysql_num_rows` returns how many rows are in the results
  - if it returns 0, no data matched the query
  - if it returns 1, a single record (row) matched the query
  - if it returns > 1, many records matched the query

Other MySQL PHP functions

- `mysql_num_fields` : returns number of columns per result
- `mysql_list_dbs` : returns a list of databases on this server
- `mysql_list_tables` : returns a list of tables in current database
- `mysql_list_fields` : returns a list of fields in the current data
- complete list

IMDb database

Actor

id	fname	lname	gender
433259	William	Shatner	M
797926	Britney	Spears	F
831289	Sigourney	Weaver	F
...			

Movie

id	name	year
112290	Fight Club	1999
209658	Meet the Parents	2000
210511	Memento	2000
...		

Cast

aid	mid	Role
433259	313398	Capt. James T. Kirk
433259	407323	Sgt. T.J. Hooker
797926	342189	Herself
...		

- database name is `imdb` on server `webster.cs.washington.edu`
- other tables:
- Director (id, fname, lname)
- Movie_Director (did, mid)
- Movie_Genre (mid, genre)

Practice problem: Movie search

- Write a PHP script that connects to the `imdb` database on `webster` and searches for all movies whose names match a given prefix, displaying them as an HTML table. Assume that the prefix is a query string parameter passed into the script.
- Consider modifying the code so that, if only one movie matches, it will print the IDs of all actors who acted in that movie. (This isn't very useful, but we'll improve it next time.)

Combining multiple tables: cross product

```
SELECT column(s) FROM table1, table2, ..., tableN;
```

```
SELECT * FROM Student, Grades;
```

- cross product : combines each row of first table with each row of second
- achieved in SQL by specifying multiple tables in `FROM` clause of `SELECT` statement
- produces *M* * *N* rows, where table 1 has *M* rows and table 2 has *N*
- you probably don't want to do this (too much irrelevant data)

Cross product example

```
SELECT * FROM Student, Grades;
```

name	SID	email	SID	CID	grade
142	Bart	bart@fox.com	142	CPS116	B-
142	Bart	bart@fox.com	142	CPS114	B
142	Bart	bart@fox.com	123	CPS116	B+
142	Bart	bart@fox.com	857	CPS116	A+
142	Bart	bart@fox.com	857	CPS130	A+
142	Bart	bart@fox.com	456	CPS114	C
123	Milhouse	milhouse@fox.com	142	CPS116	B-
123	Milhouse	milhouse@fox.com	142	CPS114	B

... (24 rows returned)

- much of the data is meaningless (e.g. Bart mixed with Lisa's courses)
- some columns might repeat (SID)

Joins

```
SELECT column(s) FROM table1, table2, ..., tableN WHERE condition(s);

SELECT column(s) FROM table1
JOIN   table2 ON condition(s)
...
JOIN   tableN ON condition(s);

SELECT * FROM Student, Grade WHERE Student.SID = Grade.SID;
```

- join : a relational database operation that combines records from two or more tables if they satisfy certain conditions
- often the rows are linked by key column values

Join example

```
SELECT * FROM Student, Grade WHERE Student.SID = Grade.SID;
```

name	SID	email	SID	CID	grade
123	Milhouse	milhouse@fox.com	123	CPS116	B+
142	Bart	bart@fox.com	142	CPS116	B-
142	Bart	bart@fox.com	142	CPS114	B
456	Ralph	ralph@fox.com	456	CPS114	C
857	Lisa	lisa@fox.com	857	CPS116	A+
857	Lisa	lisa@fox.com	857	CPS130	A+

- `table.column` disambiguates two columns with the same name
- an equivalent query:

```
SELECT * FROM Student JOIN Grade ON Student.SID = Grade.SID;
```

Filtering columns during a join

```
SELECT name, Grade.* FROM Student, Grade
WHERE  Student.SID = Grade.SID;
```

name	SID	CID	grade
Milhouse	123	CPS116	B+
Bart	142	CPS116	B-
Bart	142	CPS114	B
Ralph	456	CPS114	C
Lisa	857	CPS116	A+
Lisa	857	CPS130	A+

- if a column name only exists in one table, it may be written by itself
- to specify all columns from a table, write `table.*`

Giving names to tables

```
SELECT name, g.*
FROM   Student s, Grade g
WHERE  s.SID = g.SID;
```

name	SID	CID	grade
Milhouse	123	CPS116	B+
Bart	142	CPS116	B-
Bart	142	CPS114	B
Ralph	456	CPS114	C
Lisa	857	CPS116	A+
Lisa	857	CPS130	A+

- can give optional names to tables or columns, like a variable name in Java

Self-joins

```
SELECT *
FROM   Student s1, Student s2
WHERE  s1.SID < s2.SID;
```

name	SID	CID	grade
Milhouse	123	CPS116	B+
Bart	142	CPS116	B-
Bart	142	CPS114	B
Ralph	456	CPS114	C

Lisa	857	CPS116	A+
Lisa	857	CPS130	A+

- can give optional names to tables or columns, like a variable name in Java

Practice problem: Cast list for a movie

Write a PHP script that, when given a movie, shows the names of all female actors that appeared in it. (To do this, you will need to perform an SQL query with join operations.)