Databases and SQL

CSE 190 M (Web Programming) Spring 2008
University of Washington

References: SQL syntax reference, w3schools tutorial

Lecture outline

- relational database concepts
- Structured Query Language (SQL)
- using databases in PHP

Relational database concepts

What is a database, and how does it work?

Relational databases

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

Why use a database?

- **powerful**: can search it, filter data, combine data from multiple sources
- **fast**: can search/filter a database very quickly compared to a file
- **big**: scale well up to very large data sizes
- **safe**: built-in mechanisms for failure recovery (e.g. **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
- Microsoft SQL Server (powerful) and Microsoft Access (simple)
- PostgreSQL (powerful/complex free open-source database system)
- SQLite (transportable, lightweight free open-source database system)
- MySQL (simple free open-source database system)
    - Many servers run "LAMP" (Linux, Apache, MySQL, and PHP)
    - Wikipedia is run on PHP and MySQL
    - we will use MySQL in this course

World Database

Countries

code	name	continent	independance_year	population	gnp	head_of_state	...
AFG	Afghanistan	Asia	1919	22720000	5976.0	Mohammad Omar	...
NLD	Netherlands	Europe	1581	15864000	371362.0	Beatrix	...
...	...	...	...	...	...	...	...

Other columns: **region**, **surface_area**, **life_expectancy**, **gnp_old**, **local_name**, **government_form**, **capital**, **code2**

Cities

id	name	country_code	district	population
3793	New York	USA	New York	8008278
1	Los Angeles	USA	California	3694820
...	...	...	...	...

CountriesLanguages

country_code	language	official	percentage
AFG	Pashto	T	52.4
NLD	Dutch	T	95.6
...	...	...	...

Structured Query Language (SQL)

the standard language for interacting with a database

SQL basics

```
SELECT name FROM Cities WHERE id = 17;
INSERT INTO Countries VALUES ('SLD', 'ENG', 'T', 100.0);
```
SQL

- 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;
```
SQL

```
SELECT name, code FROM Countries;
```
SQL

name	code
China	CHN
United States	IND
Indonesia	USA
Brazil	BRA
Pakistan	PAK
...	...

- 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

- SSH to Webster, then type:

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

mysql> USE world;
Database changed

mysql> SELECT * from Cities;
+----+---------------+-------------+----------+------------+
| id | name          | country_code | district | population |
+----+---------------+-------------+----------+------------+
|  1 | Kabul         | AFG          | Kabol    |    1780000 |
|  2 | Qandahar      | AFG          | Qandahar |     237500 |
|  3 | Herat         | AFG          | Herat    |     186800 |
|  4 | Mazar-e-Sharif | AFG          | Balkh    |     127800 |
...
+----+---------------+-------------+----------+------------+
```

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

The DISTINCT modifier

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

```
SELECT language
FROM CountriesLanguages;
```

language
Dutch
English
English
Papiamento
Spanish
Spanish
Spanish
...

```
SELECT DISTINCT language
FROM CountriesLanguages;
```

language
Dutch
English
Papiamento
Spanish
...

- eliminates duplicates from the result set

The `WHERE` clause

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

```
SELECT name, population FROM Cities WHERE country_code = "FSM";
```
SQL

name	population
Weno	22000
Palikir	8600

- `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)
```
SQL

```
SELECT name, gnp FROM Countries WHERE gnp > 2000000;
```
SQL

code	name	gnp
JPN	Japan	3787042.00
DEU	Germany	2133367.00
USA	United States	8510700.00
...	...	...

- 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

```sql
SELECT * FROM Cities WHERE code = 'USA' AND population >= 2000000;
```

id	name	country_code	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
...	...	...	...	...

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

Approximate matches: `LIKE`

```sql
WHERE column LIKE pattern
```

```sql
SELECT code, name, population FROM Countries WHERE name LIKE 'United%';
```

code	name	population
ARE	United Arab Emirates	2441000
GBR	United Kingdom	59623400
USA	United States	278357000
UMI	United States Minor Outlying Islands	0

- `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 by a column: `ORDER BY`

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

```
SELECT code, name, population FROM Countries
WHERE name LIKE 'United%' ORDER BY population;
```
SQL

code	name	population
UMI	United States Minor Outlying Islands	0
ARE	United Arab Emirates	2441000
GBR	United Kingdom	59623400
USA	United States	278357000

- can write ASC or DESC to sort in ascending (default) or descending order:

  ```
  SELECT * FROM Countries ORDER BY population DESC;
  ```
  SQL

- can specify multiple orderings in decreasing order of significance:

  ```
  SELECT * FROM Countries ORDER BY population DESC, gnp;
  ```
  SQL

Using a database in PHP

PHP code on your server that can access database data

Complete PHP MySQL example

```
# connect to world database on local computer
$db = mysql_connect("localhost", "traveler", "packmybags");
mysql_select_db("world");

# execute a SQL query on the database
$results = mysql_query("SELECT * FROM Countries WHERE population > 100000000

# loop through each country
while ($row = mysql_fetch_array($results)) {
?>

  <li> <?= $row["name"] ?>, ruled by <?= $row["head_of_state"] ?> </li>

<?php
}
?>
```
PHP

Connecting to MySQL: `mysql_connect`

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

```
# connect to world database on local computer
$db = mysql_connect("webster.cs.washington.edu", "traveler", "packmybags");
mysql_select_db("world");
```
PHP

- `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

Performing queries: `mysql_query`

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

$results = mysql_query("SQL query");
...
```

```php
$results = mysql_query("SELECT * FROM Cities WHERE code = 'USA'
                        AND population >= 2000000;");
```

- `mysql_query` sends a SQL query to the database
- returns a special result-set object that you don't interact with directly, but instead pass to later functions
- SQL queries are in `" "`, end with `;`, and nested quotes can be `'` or `\"`

Result rows: `mysql_fetch_array`

```php
$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_fetch_array` returns one result row as an associative array
  - the column names are its keys, and each column's values are its values
  - example: `$row["population"]` gives the population from that row of the results

Error-checking: `mysql_error`

```php
$db = mysql_connect("webster.cs.washington.edu", "traveler", "packmybags");
if (!$db) {
  die("SQL error occurred on connect: " . mysql_error());
}
if (!mysql_select_db("world")) {
  die("SQL error occurred selecting DB: " . mysql_error());
}
$query = "SELECT * FROM Countries WHERE population > 100000000;";
$results = mysql_query($query);
if (!$results) {
  die("SQL query failed:\n$query\n" . mysql_error());
}
```

- SQL commands can fail: database down, bad password, bad query, ...
- for debugging, always test the results of PHP's `mysql` functions
  - if they fail, stop script with `die` function, and print `mysql_error` result to see what failed
  - give a descriptive error message and also print the query, if any

Complete example w/ error checking

```php
# connect to world database on local computer
$db = mysql_connect("localhost", "traveler", "packmybags");
if (!$db) {
  die("SQL error occurred on connect: " . mysql_error());
}
if (!mysql_select_db("world")) {
  die("SQL error occurred selecting DB: " . mysql_error());
}

# execute a SQL query on the database
$query = "SELECT * FROM Countries WHERE population > 100000000;";
$results = mysql_query($query);
if (!$results) {
  die("SQL query failed:\n$query\n" . mysql_error());
}

# loop through each country
while ($row = mysql_fetch_array($results)) {
?>
  <li>
    <?= $row["name"] ?>, ruled by <?= $row["head_of_state"] ?>
  </li>
<?php
}
?>
```

PHP

Other MySQL PHP functions

- `mysql_num_rows` : returns number of rows matched by the query
- `mysql_num_fields` : returns number of columns per result in the query
- `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