Introduction to Databases

Class 28

Data Storage

- there's a difference between
  - data storage
  - data transfer
- we have already discussed transfer:
  plain text, XML, JSON, pre-formatted HTML
- data can be stored in ASCII files formatted as
  - CSV (e.g., the GRE word list)
  - JSON (no examples yet)
  - XML (e.g., the Gettysburg Address)
- or stored in a database management systems (DBMS)

Why Databases

ASCII file storage is great for

- systems with few users
- read-only data
- small amounts of data
- simple organization of data

a DBMS is required for:

- large amounts of data
- systems with many users, especially if there are race conditions or access controls
- complex relationships among data

Features

a DBMS provides:

power	search, filter, combine data from multiple sources
speed	search, sort, and filter much faster than plain text
scaling	up to very large data sizes
safety	mechanisms for failure recovery
consistency	mechanisms to keep data uncorrupted
security	mechanisms for access control
concurrency	features let multiple users work simultaneously without conflict, managing race conditions
abstraction	layer between data and applications

DBMS Types

Type	**Description**
Hierarchical	historical; used exclusively with COBOL
RDBMS	by far the most common; table-based relational schema
OODBMS	conceptually powerful, but rarely used
NoSQL	flexible data format, no fixed schema

- we will only use RDBMS

RDBMS Systems

common systems are

- Oracle: the 800 lb gorilla, very expensive
- MS SQL Server: not free, not fully standards-compliant
- SQLite: extremely lightweight free open-source system, installed on every Android and Apple i-device
- MySQL: most widely used desktop system; owned by Oracle; free to use, almost fully standards-compliant
  - the DB part of the Linux-Apache-MySQL-PHP LAMP stack
- MariaDB: non-Oracle, free version of MySQL ("mysql" on sand)
- PostgreSQL: free, fully standards-compliant, not as widely used

Login

on sand (this only works from sand):

```
$ mysql -p -h borax.truman.edu -u abc1234
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 306
Server version: 8.0.28-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

MySQL [(none)]>
```

- your password is the last 4 digits of your banner followed by the digits in your username

Choose Database

everyone in class has access to three databases

- one with the same name as your username (e.g., abc1234)
- imdb
- world

you specify which database you wish to use:

```
MySQL [(none)]> use imdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [imdb]>
```

Getting Information

```
MySQL [imdb]> show tables;
+-----------------+
| Tables_in_imdb |
+-----------------+
| actor           |
| director        |
| director_genre |
| movie           |
| movie_director |
| movie_genre    |
| role            |
+-----------------+
7 rows in set (0.002 sec)
```

Getting Information

```
MySQL [imdb]> describe actor;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | int unsigned | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(255) | NO   |     | NULL    |                |
| last_name  | varchar(255) | NO   |     | NULL    |                |
| sex        | char(1)      | YES  |     | NULL    |                |
| film_count | int unsigned | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
5 rows in set (0.003 sec)
```

Getting Information

```
MySQL [imdb]> show create table actor;
+-------+----------------------------------------------------------------+
| Table | Create Table                                                   |
+-------+----------------------------------------------------------------+
| actor | CREATE TABLE `actor` (                                         |
|       | `id` int unsigned NOT NULL AUTO_INCREMENT,                     |
|       | `first_name` varchar(255) NOT NULL,                            |
|       | `last_name` varchar(255) NOT NULL,                             |
|       | `sex` char(1) DEFAULT NULL,                                    |
|       | `film_count` int unsigned DEFAULT NULL,                        |
|       | PRIMARY KEY (`id`)                                             |
|       | ) ENGINE=InnoDB AUTO_INCREMENT=841406 DEFAULT CHARSET=utf8     |
+-------+----------------------------------------------------------------+
1 row in set (0.064 sec)
```

SQL

- Structured Query Language
- the language of RDBMS

- a declarative language that describes the data to find,
- as opposed to a procedural language that says how to find it

SQL

SQL provides statements for:

data queries	retrieve, add, change, delete data
transactions	to ensure that data remains consistent
data definition	create tables, alter their structure, drop them
admin	set permissions, etc.

Select

Get data from a table

```
mysql> select last_name from director limit 5;
+-----------+
| last_name |
+-----------+
| Adamson   |
| Aronofsky |
| Braff     |
| Cameron   |
| Clements  |
+-----------+
5 rows in set
```

Filter

Get data from a table that match some criteria

```
mysql> select first_name, last_name from director
         where last_name = 'Coppola';
+--------------+-----------+
| first_name   | last_name |
+--------------+-----------+
| Francis Ford | Coppola   |
| Sofia        | Coppola   |
+--------------+-----------+
2 rows in set (0.00 sec)
```

Filter

```
mysql> select name, year
       from movie
       where ranking > 8 and year = 2003;
+----------------------+------+
| name                 | year |
+----------------------+------+
| Kill Bill: Vol. 1    | 2003 |
| Mystic River         | 2003 |
+----------------------+------+
2 rows in set (0.00 sec)
```

Where Operators

The where clause can use these operators:

- `= > >= < <= <>`
- `BETWEEN min AND max`
- `LIKE pattern` wildcard is %, not *
- `IN (value, value, ..., value)` set operator
- NOT
- OR
- AND

Order, Limit

In order, ascending or descending:

```
mysql> select name from movie
       order by name desc
       limit 7;
+-----------------+
| name            |
+-----------------+
| Vanilla Sky     |
| UHF             |
| Titanic         |
| Stir of Echoes  |
| Star Wars       |
| Snatch.         |
| Shrek           |
+-----------------+
```

Aggregating

```
mysql> select avg(ranking) as rankavg
       from movie where year >= 2000;
+--------------------+
| rankavg            |
+--------------------+
| 7.766666730244954 |
+--------------------+
1 row in set

mysql> select count(*) as cnt
       from movie
       where year between 1980 and 2000;
+-----------+
| cnt       |
+-----------+
|        23 |
+-----------+
1 row in set
```

Group By

```
mysql> select year, format(avg(ranking), 2) as avg
         from movie
         where year between 1990 and 1995
         group by year order by year;
+------+------+
| year | avg  |
+------+------+
| 1991 | 7.80 |
| 1992 | 7.90 |
| 1994 | 8.85 |
| 1995 | 7.90 |
+------+------+
4 rows in set (0.00 sec)
```

Group By and Having

```
mysql> select year, format(avg(ranking), 2) as average
         from movie
         where year between 1990 and 1995
         group by year having average > 8;
+------+---------+
| year | average |
+------+---------+
| 1994 | 8.85    |
+------+---------+
1 row in set
```

Order of Clauses

SQL requires the following order:

1. select
2. from
3. where
4. group by
5. having
6. order by
7. limit

All Columns

For testing and debugging, you can easily select all columns:

```
mysql> select * from movie where name like '%x%';
+--------+-------------+------+------+
| id     | name        | year | rank |
+--------+-------------+------+------+
| 207992 | Matrix, The | 1999 |  8.5 |
+--------+-------------+------+------+
1 row in set
```

But don't ever use select * in a production system!