Today …
- Quiz 4
- SQL (intro)

Homework
- HW 4 out
Connecting to the DB Server

See the db-server info sheet on the course webpage ...

1. `ssh ada.gonzaga.edu` (establishes connection)
2. `mysql -h cps.database` (establishes connection)

In (1):
- login to server via port 22 (ssh)
- interact with Linux (terminal) shell

In (2):
- login to server via port 3306 (mysql)
- interact with the SQL shell (for MariaDB)

* Currently: from on-campus can bypass ada and connect directly to eps database (requires installing mysql client program)

Some prefer to use MySQL workbench ...

- similar idea, just a GUI “veneer”
- better to learn using the command line, not the workbench
Structured Query Language (SQL)

The language used to “talk to” the DBMS

SQL can be used for many operations ...

• To create tables

```
CREATE TABLE account (  
   acct_num INT UNSIGNED,  
   owner VARCHAR(50),  
   balance DECIMAL(8,2) NOT NULL,  
   acct_type VARCHAR(8),  
   PRIMARY KEY (acct_number)
);
```

• To query the database

```
SELECT *  
FROM account  
WHERE type = "checking";
```

• To insert rows into a table

```
INSERT INTO account VALUES (106, "Alice", 1000, 10);
```

• And so on ...
More on SQL

SQL is a standard

- There have been a series: 1986, 1989, 1992 (SQL 2), ... 2016 (SQL:2016)

Even though it is a standard

- DBMS products differ in how much they support
- And many implement extra features (extensions)

SQL is considered a declarative language

- In general, this means that you say what you want to happen
- Not how to perform it (the DBMS figures it out)

SQL is largely case insensitive

- Various conventions in use (lowercase, uppercase, camel case, etc.)
- Often keywords in uppercase, ids in lowercase
- Some systems allow for case-sensitive names (at least partially)
Declaring Tables in SQL (MariaDB/MySQL)

Recall that schema attributes have “**domains**” — data types + constraints

The Basic SQL Data Types (Supported by MariaDB/MySQL)

**Integer values:**
- `INT` or `INTEGER` 4 bytes
- `MEDIUMINT` 3 bytes (not SQL)
- `SMALLINT` 2 bytes
- `TINYINT` 1 byte (not SQL)
- `BIGINT` 8 bytes (not SQL)
- Also `UNSIGNED` (e.g., `INT UNSIGNED`)

**Floating-point values:**
- `FLOAT` 4 bytes
- `DOUBLE` 8 bytes
- `FLOAT(M, D)` and `DOUBLE PRECISION(M, D)`
  - *values up to $M$ digits of which $D$ may be after the decimal point*
  - *values (with more precision) are rounded*
- Also supports “exact” floating point types (e.g., `DECIMAL(8, 2)`)

**Boolean and Binary values:**
- `BOOL` or `BOOLEAN` 0 is false, not 0 is true
- `BIT(M)` $M$ binary digits (bits)
String values:

- **CHAR(N)**  \[0 \leq N \leq 255\] characters (fixed length)
- **VARCHAR(N)**  \[0 \leq N \leq 65,535\] characters (variable length)
- **ENUM(`v1', `v2', ...)** one of given string values
- **BLOB**
  * "binary large object"
  * can store a variable amount of data (variable length)
  * stored as byte strings (no character set)
  * also TINYBLOB, MEDIUMBLOB, LONGBLOB (different max lengths)
- **TEXT** same as blob for character data
- **TINYTEXT** up to 255 characters

Date and Time values:

- **YEAR** YYYY (or `YYYY') format from 1901 to 2155 (and 0000)
- **DATE** `YYYY-MM-DD' format
- **TIME** multiple formats, e.g., `HH:MM' and `HH:MM:SS'
- **DATETIME** both a date and a time, multiple formats

Others:

- various other string and numeric types
- geometric types
- JSON values
- **SET** types (e.g., SET(`value1', `value2', ...))