Today

- Quiz 1
- Intro to SQL
- Creating tables in SQL

Assignments

- HW1 due
- HW2 out, due next Tues
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 (
    number INT NOT NULL,
    owner VARCHAR(50),
    balance DOUBLE,
    type VARCHAR(8),
    PRIMARY KEY (number)
);
```

- To query the database

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

- To insert rows into a table

```
INSERT INTO account VALUES (106, "M. Cruz", 10);
```

- And so on ...
More on SQL

SQL is a standard

- There have been a series: 1986, 1989, 1992 (SQL 2), ... 2016 (SQL:20016)

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

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
Declaring Tables in SQL (MariaDB/MySQL)

Recall that schema attributes have “domains” — data types

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

Plus others (e.g., for Date and Time values)
Basic Table Declarations (CREATE TABLE)

```
CREATE TABLE tablename (
    att1 type1,
    att2 type2,
    ...
);
```

where:

- `att1` is the name of the attribute and `type1` is the data type (domain)
- `table` created from within an existing database

We can also add various additional information:

- constraints (e.g., NOT NULL, keys, foreign keys)
- default values
- in MySQL, the underlying DB engine
- and so on ...
Defining the Accounts table:

- account(number, owner, balance, type)

Q: What would be reasonable data types?

Q: What is the CREATE TABLE statement?

```sql
CREATE TABLE account (
    number INT,
    owner VARCHAR(50), -- or: owner TINYTEXT
    balance DECIMAL(8,2),
    type VARCHAR(8), -- or: type ENUM('checking', 'savings')
);
```

Recall that “number” is a primary key ...

```sql
CREATE TABLE account (
    number INT,
    owner VARCHAR(50),
    balance DECIMAL(8,2),
    type VARCHAR(8),
    PRIMARY KEY (number)
);
```
Q: How would we create this table? ... ignoring FKs for now

- check(account, number, amount, date)

```
CREATE TABLE check (
    account INT,
    check_number INT,
    date VARCHAR(10), /* better to use date type here */
    amount DECIMAL(8,2),
    PRIMARY KEY (account, check_number)
);
```

**Defining a foreign key:**

```
CREATE TABLE check (
    account INT,
    check_number INT,
    date VARCHAR(10),
    amount DECIMAL(8,2),
    PRIMARY KEY (account, check_number),
    FOREIGN KEY (account) REFERENCES accounts (number)
);
```