Databases

CMSE 890-602

When is a database not a table?

- A database is a *collection* of tables
- A relational database includes *relationships* between tables
- Relationships are defined with *keys*
- Keys are unique identifiers for rows
    - Primary key in the originating table
    - Foreign key when a primary key is linked to another table

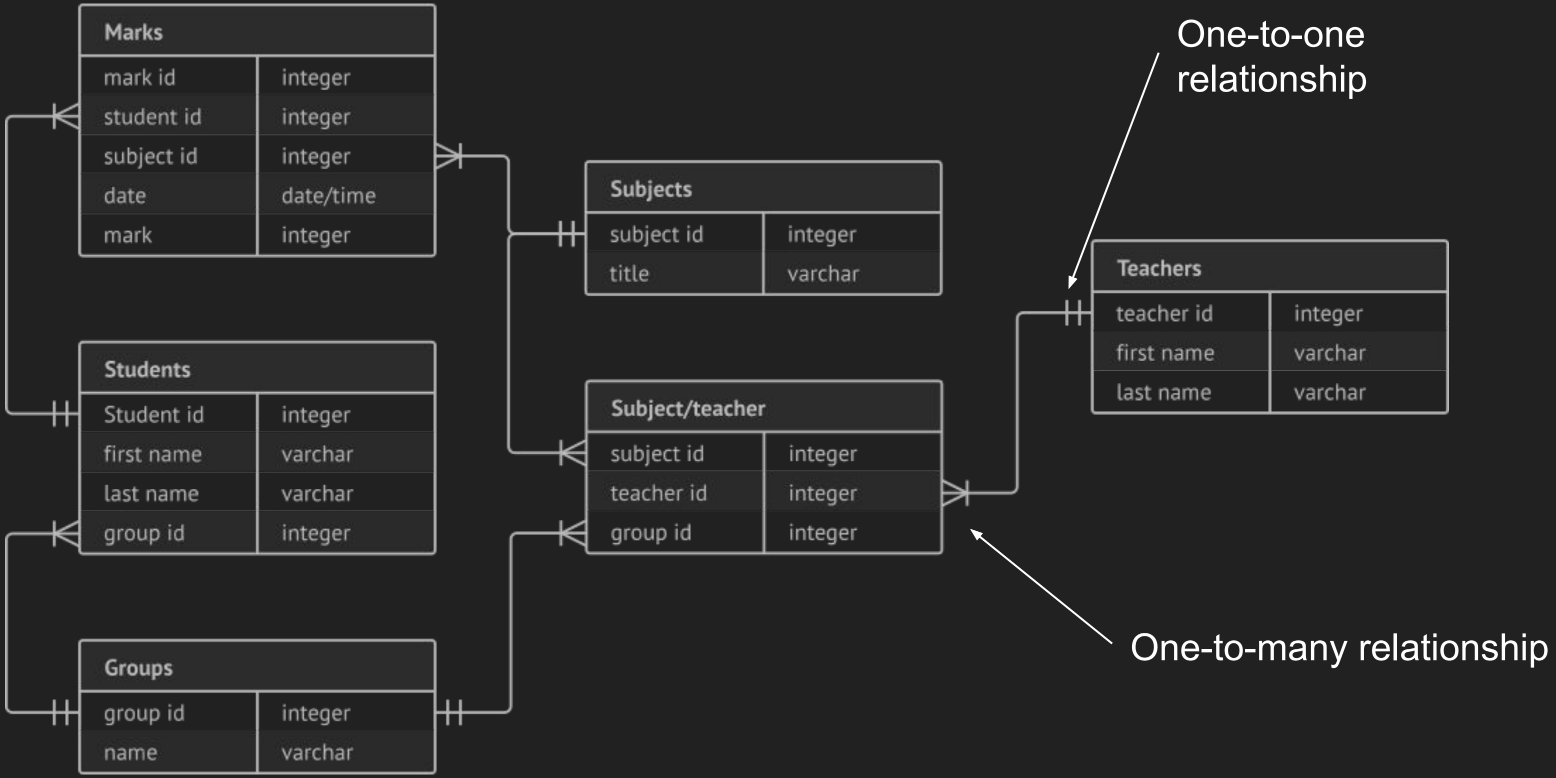
Relationship Types

- One-to-one: there can only be one instance of the key in each related table
- One-to-many: there can be multiple instances of the keys in one of the related tables
- Many-to-many: there can be multiple instances of the keys in both of the related tables

Why use a database?

- Great for data with relationships to other data
- Monolithic (i.e. one file to worry about)
- Easy to query across multiple tables at once

Entity relationship diagrams



Cursors and Transactions

- A *cursor* is a process that enables database access
- They “point” to each row to process commands
- Cursors can require additional overhead compared to other methods

- A *transaction* is any unit of work done on a database, such as a command
- Transactions should pass the “ACID test”

ACID test

- Atomicity (transactions are single units),
- Consistency (database state preserves constants),
- Isolation (transactions can happen simultaneously or sequentially with no difference in result),
- Durability (completed transactions are guaranteed to apply even with system failure)

SQL

- Structured Query Language
- Designed to be simple and logical
- Query = collection of requests to the database to return data
- Database format
- **Requires a server to function** in most implementations
  - This means you can host a database on the web for remote access
  - High security
  - High scalability
  - Reliable (if you put the work in)

SQLite

- An *implementation* of SQL written in C that can be used to query SQL databases
- Supports multiple interface languages e.g. Python, Java
- Does **NOT** require a server to run
- Supports all major SQL commands
- Can load databases into memory (fast!)
- Dynamic types for database columns, so any type in any column
    - Something to be careful about!

Basic syntax and commands

COMMAND *argument* ADDITIONAL *argument*
ADDITIONAL *argument;*

SELECT *column1, column2* FROM *table1, table2*
WHERE *condition;*

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

WHERE Conditions

- AND, OR, NOT
- LIKE ‘string%’
  - % is a wildcard character that allows for matching, in this case “string” with any characters after
- IN (‘entry1’, ‘entry2’, ‘entry3’)
- BETWEEN value1 AND value2

Other useful commands

```
CREATE DATABASE databasename;

CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
  ....
);
```

In-class assignment

- Go to https://mystery.knightlab.com/walkthrough.html
- Go through the steps
- Try to solve the mystery!
- You can also go to https://github.com/NUKnightLab/sql-mysteries
- This goes through installing a local SQLite client to solve the mystery
- Submit screenshots of your SQL queries and results to the D2L assignment
- Please annotate your queries with comments in /* */ blocks so I know your working

Homework

- Python 3 has a built-in SQLite module, sqlite3
    - https://docs.python.org/3/library/sqlite3.html
- R has a SQLite package: https://rsqlite.r-dbi.org/
- Using one of these packages, convert your in-class assignment into code
- **Also complete the optional task of finding the mastermind behind the murder**
- Upload the script to D2L
- As usual, please comment if you use AI to get the code done. Make sure it works!

Pre-class

Go to https://snakemake.readthedocs.io/en/stable/tutorial/tutorial.html

Complete the "Basics" tutorial as far as you can. Upload (to D2L) the SVG files you create for the DAGs, and the histogram plot if you manage to get that far. Write a sentence or two about how far you got with supporting screenshot(s). 3/4 points for installing snakemake properly and giving it a good try, 4 points for reaching Step 6.

Strongly recommended to use your ICER account for this if you have one. Please contact me if you do not, and I can have one created for you.

If you have a Windows machine and need a Linux environment, use your ICER account if you have one; otherwise, the tutorial provides the setup in a cloud-based gitpod environment (requires connection to your github account and answering a few questions, with limited runtime available). The tutorial also provides information about using Windows Subsystem for Linux as an alternative.