

# Discussion 2

More ER Diagrams, Project 1 Intro  
EECS 484

# Logistics

- Homework 1 due Sept. 15, 11:55 PM Eastern
- Project 1 due Sept. 22, 11:55 PM Eastern
  - Will require having a SQLPlus account on CAEN
    - Please let us know early if you don't have an account
    - Instructions to connect to CAEN and SQLPlus (Discussion 2 slides and Project 1 pdf)
- Midterm time: Oct. 21st 7:30 - 9:30 PM Eastern
  - Send an email to [eeecs484staff@umich.edu](mailto:eeecs484staff@umich.edu) if you have a time conflict

# More ER Diagrams

# Key constraints in a ternary relationship

only one combo e.g. (D1, L1) can relate to E1

Each employee works in **at most one** combination of department and location.

Example: either both null, or neither null.

1. (E1, D1, L1)

2. ~~(E1, D1, L1)~~

3. ~~(E1, D2, L1)~~

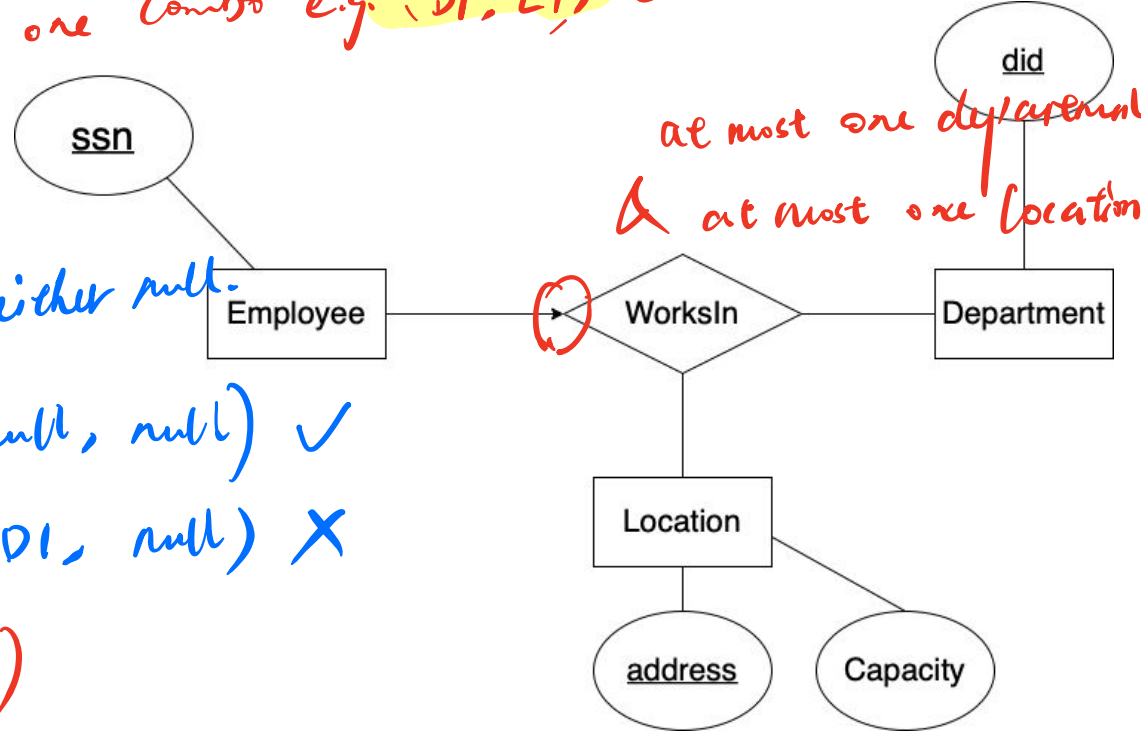
4. ~~(E1, D1, L2)~~

5. (E2, D1, L1)

(E1, null, null) ✓

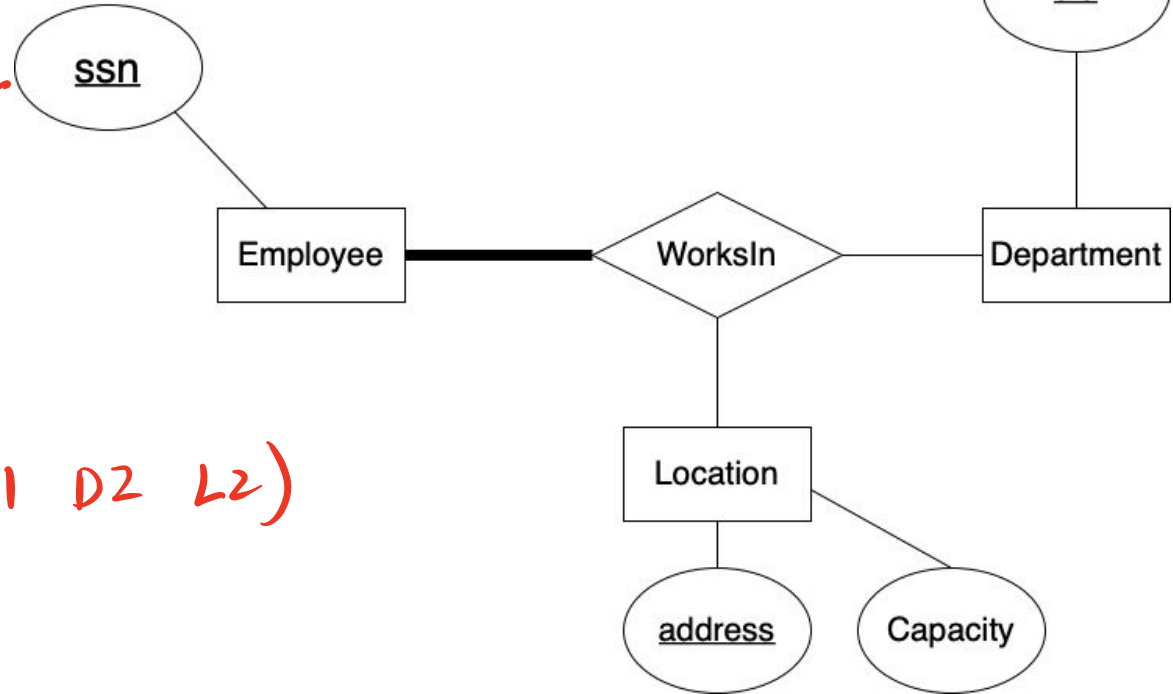
(E1, D1, null) ✗

repetitive  
repetitive )



# Participation constraints in a ternary relationship

Each employee works in **at least one** combination of department and location.



Example:

1. E has E1, E2
2. (E1, D1, L1)
3. Need (E2, Dx, Ly)

(E1 D2 L2)

# ER diagram practice

Professors have a unique SSN, a name, an age, a rank, and a research specialty.

Projects have a unique project number, a sponsor name, a starting date, an ending date, and a budget.

*key.*

Graduate students have a unique SSN, a name, an age, and a degree program (e.g., M.S. or Ph.D.).

*← exactly one (key + participation).*

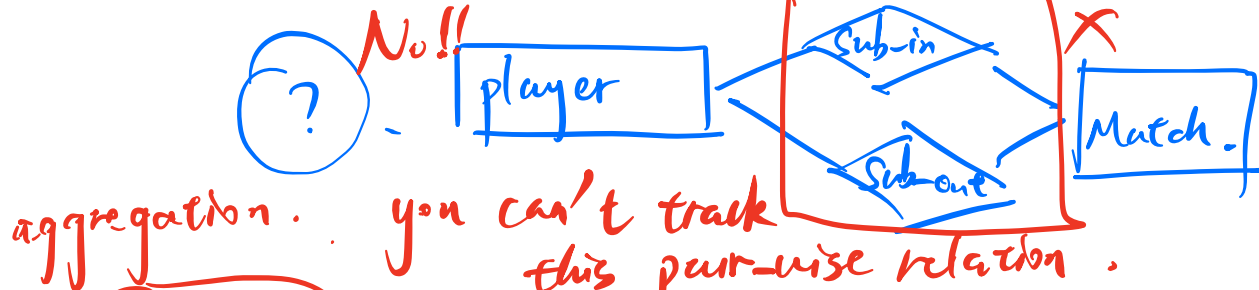
Each project is managed by one professor (known as the project's principal investigator) and is worked on by one or more professors (known as the project's co-investigators).

*per constraint.*

Professors can manage and/or work on multiple projects.

Each project is worked on by one or more graduate students (known as the project's research assistants).

# ER diagram practice



When graduate students work on a project, their work on the project must be supervised by exactly one professor. Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.

Departments have a unique department number, a department name, and a main office.

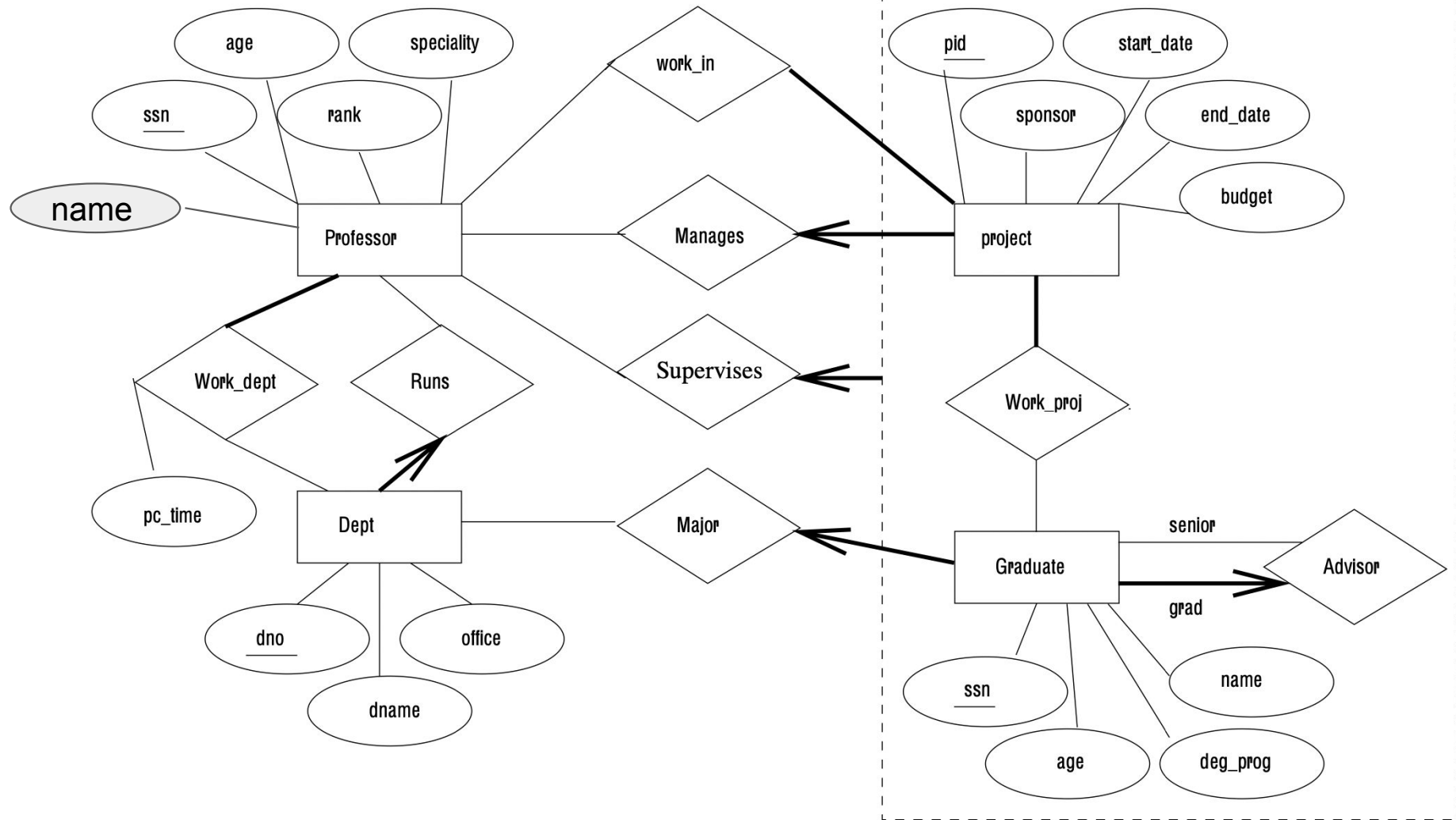
Departments must have one professor (known as the chairman) who runs the department.

Professors work in one or more departments, and for each department that they work in, a time percentage is associated with their job.

Graduate students have one major department in which they are working on their degree.

Each graduate student has exactly one more senior graduate student (known as a student advisor) who advises him or her what courses to take.







# Project 1 Setup

# Helpful Tools

- We're going to be using SQLPlus for the next two projects
  - Tool to connect to Oracle Databases
  - Need to connect to CAEN to be able to use the SQLPlus CLI
- Requires CAEN account
  - CoE students have default
  - If you do not have one go [here](#)

# SSH + FTP

- SSH (Secure Shell)
  - Linux and Mac users will have SSH built into their terminals
  - Windows users can install Windows Subsystem for Linux (WSL) to use SSH (recommended)
  - Can install CAEN VNC Client instead
- FTP (File Transfer Protocol) and SCP (Secure Copy Protocol)
  - Ways to upload files from your local machine to a server
  - Not necessary if you do all of your development on CAEN
- Need to connect to `login.engin.umich.edu` for both SSH and FTP
  - Will need Duo

# SSH + FTP/SCP (Command line commands)

ssh [username@login.engin.umich.edu](mailto:username@login.engin.umich.edu)

scp -r [source file/dir] username@login.engin.umich.edu:[target dir]/[target name]

# SSH in VSCode

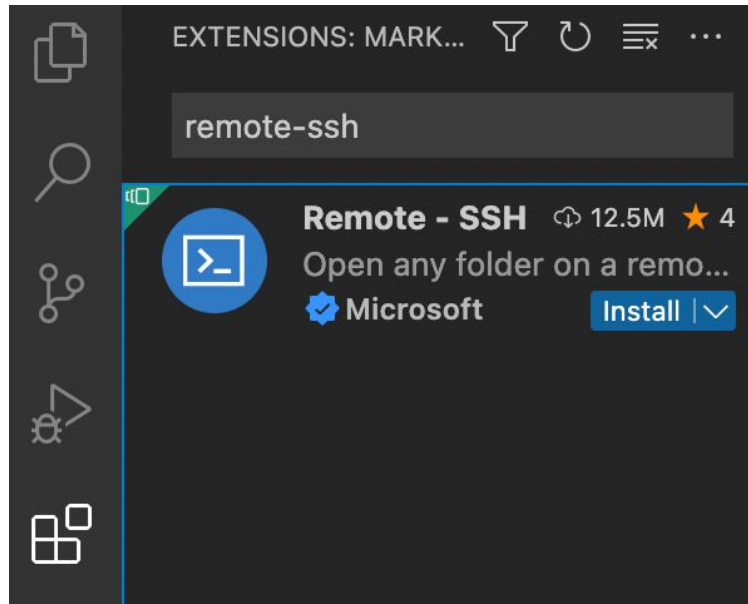
1. Create ~/.ssh/config (cd ~/.ssh -> open config or touch config)
  - a. Example:

```
# SSH multiplexing
Host *
    ControlMaster auto
    ControlPath ~/.ssh/master-%r@%h:%p

# CAEN
Host caen
    Hostname login.engin.umich.edu
    User <username>
    ControlPersist 1h|
```

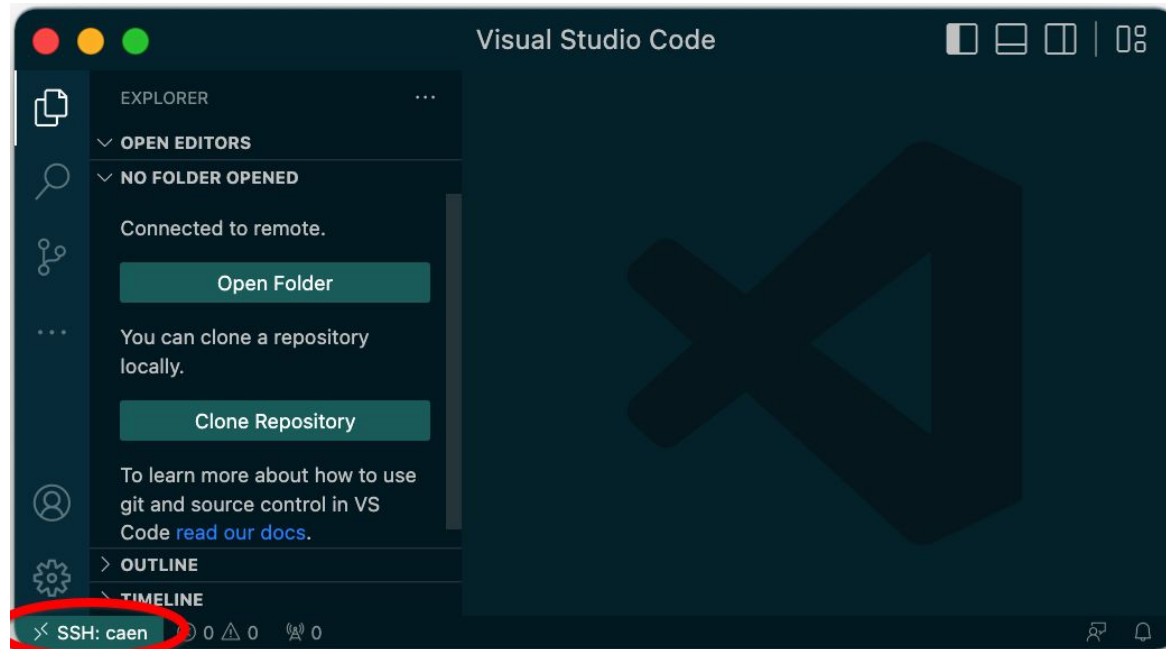
# SSH in VSCode

2. Install the Remote - SSH extension
3. Open the command palette and click Remote-SSH: Connect to Host (Command-Shift-P)
4. Choose “caen”



# SSH in VSCode

5. Enter your umich password and Duo (a new window should pop up)
6. You are now ready to remotely edit and run scripts!



# Oracle SQL

- To access the Oracle DBMS
  - SSH into CAEN or use the CAEN VNC Client
  - **module load eecs484**
    - Loads some of the tools and programs needed for this class
    - Append this to your ~/.bash\_profile
      - You don't need to type this command in every time then!
  - Launch SQLPlus with **rlwrap sqlplus**
  - Enter username/password (next slide)
  - Congrats! You can now run SQL commands!
- Documentation on the various commands [here](#)
  - Each DBMS differs slightly from each other



# Oracle SQL

- An Oracle account within the CAEN servers has been created for you
  - Username: Your username
  - Initial password: eecsclass
  - SQLPlus will prompt you to change password when you login the first time
  - **DO NOT use quotes or @ in your password**
- Make a private post on Piazza or email us at [eeecs484staff@umich.edu](mailto:eeecs484staff@umich.edu) if you don't have one for any reason
  - Adding class late, etc.

# Project 1 Overview

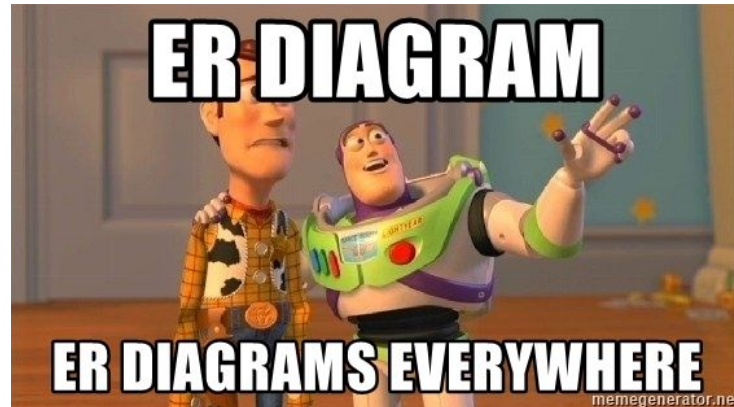
# Project 1

- You have recently been contacted by Clark Huckelburg
  - CEO of FakeBook, “the world’s fakest social media platform”™
  - Need to design a database for them to migrate to
- 4 Parts
  - Draw ER diagram
  - Translate ER diagram and specifications into relational tables
  - Populate database with existing data *INSERT INTO.*
  - Create views to make it easier to look at aggregate data
  - Best to go one part a time in order
    - some parts will need to be worked on at the same time



# Project 1 - Part 1

- Read the spec carefully!
  - Reading later parts will help you with your ER diagram
  - Draw neatly on paper and then scan or use computer tools - LucidChart, draw.io, ...
  - Try the suggested steps to creating an ER diagram (next slide)
    - Determine what's an entity, relationship, and an attribute
    - Figure out which relationships are binary vs ternary



# Creating ER Diagrams (from last week's discussion)

- Three key steps to success
  - Start with the entities and relationships
    - Make your entity squares (no attributes yet)
    - Make your relationship diamonds between squares
      - Don't worry about constraints yet
    - Handle ISA hierarchies
  - Add in attributes
    - Determine if they should belong to entity or relationship
    - Determine primary keys
  - Resolve constraints
    - Handle key and participation constraints
    - Determine what weak entities exist
    - Check relationships for potential ternary relationships

# Project 1 - Part 2

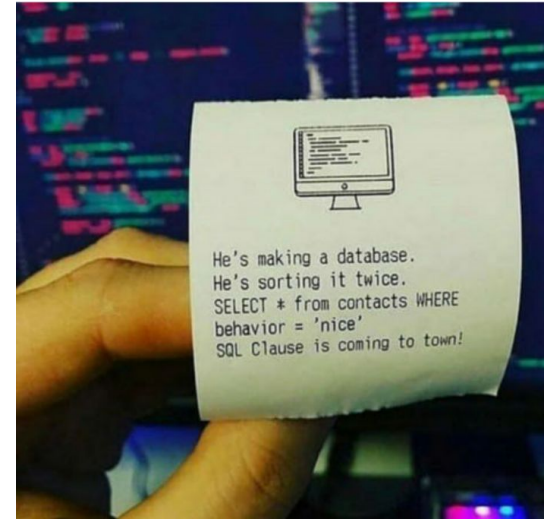
- Create the schema!
  - Turn ER diagrams into tables using SQL
    - createTables and dropTables SQL scripts
    - Will also need to create and manage triggers and sequences
  - Need to make sure all constraints are captured
    - Is it NOT NULL? UNIQUE? PRIMARY KEY? FOREIGN KEY?
- Example CREATE statement

```
CREATE TABLE Students (  
    student_id INTEGER PRIMARY KEY,  
    name VARCHAR(200) NOT NULL  
);
```

- Example DROP statement

```
DROP TABLE Students CASCADE CONSTRAINTS;
```

⇒ drop all the constraints.



# Project 1 - Part 3

insert one row:  
insert into users values ( ... )

- Populate the database

- Take public data and insert it into your tables

```
INSERT INTO Users
```

```
SELECT user_id
```

```
FROM project1.Public_User_Information;
```

- project1 is the schema in which all the data lives
- Will need to perform unions, joins, and other relational logic to insert data

# Project 1 - Part 4

- Create views

- External schema - Designed to mimic public data set

- `CREATE VIEW` Instructor\_Name `AS`  
SELECT I.last\_name  
FROM INSTRUCTOR I  
WHERE I.first\_name = 'Bob';

*table name*

- Shows only the last name from instructors where the first name is Bob

- You can check this before submitting

- `SELECT * FROM project1.Public_User_Information`  
MINUS

- `SELECT * FROM View_User_Information`

- `SELECT * FROM View_User_Information`  
MINUS

- `SELECT * FROM project1.Public_User_Information`

- Checking to make sure it is identical to the public dataset you read from

*set difference  $\Rightarrow$  empty*





# SQL

# SQL

- Structured Query Language
  - Allows you to interact with a DBMS
  - Commands differ slightly from distribution to distribution
    - All compliant languages should be about the same though
  - Other languages exist but are far less popular
  - Vulnerable to things like SQL injection
    - Way of interacting where SQL cannot tell the difference between text and code
  - Convention: All SQL keywords are uppercase
    - SQL doesn't actually care

SELECT \* FROM

Select \* From

select \* from

SeLEct \* fRoM

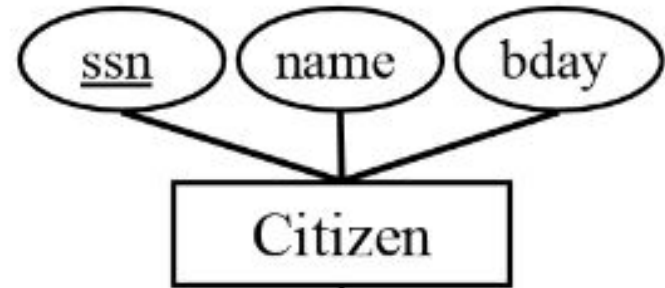
*Select \* from users where password = 'my password.'*

*my; drop table users*



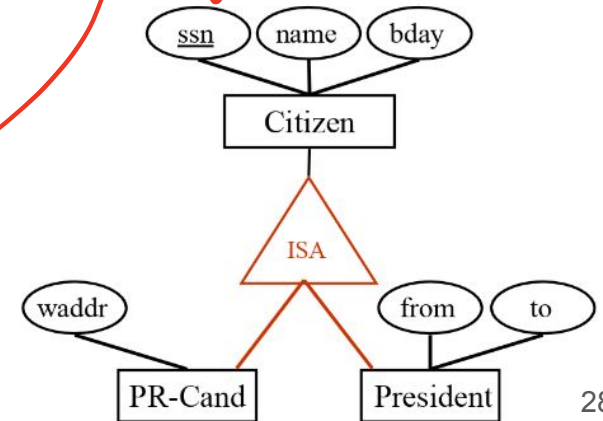
# Creating Tables

- `CREATE TABLE Table_Name (field_1 field_1_type, field_2, field_2_type ... );`
  - Makes an empty table with those columns
  - Can specify constraints too
    - Primary keys, not null, unique, foreign keys, etc.
- Example:
  - `CREATE TABLE Citizen (  
    ssn NUMBER PRIMARY KEY,  
    name VARCHAR2(100) NOT NULL,  
    bday DATETIME  
);`



# Constraints

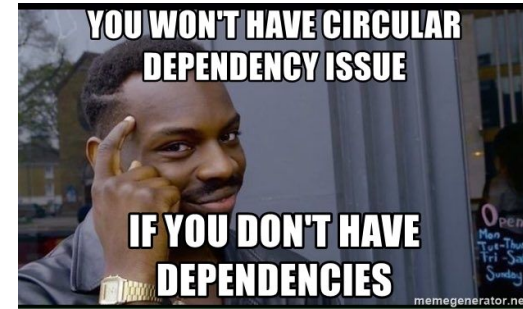
- Constraints allow the database to check the data as it's inserted
  - “Laws” of the schema that are enforced
- Examples
  - PRIMARY KEY - Necessary for every table. A unique and not null value
  - NOT NULL - Field must be populated
  - UNIQUE - Field must be unique across all rows
  - FOREIGN KEY - Links entries across tables (references primary key of another table)
  - CHECK - Makes sure data in a column meets some condition *check (age > 0)*
  - Constraints can be applied across multiple fields in the relation
  - CREATE TABLE President (
    - ssn NUMBER PRIMARY KEY,
    - name VARCHAR2(100) NOT NULL,
    - from DATETIME NOT NULL,
    - to DATETIME NOT NULL,
    - UNIQUE (from, to) );



(1, 3) ✓  
(1, 2) ✓  
(1, 2) X

# Circular Dependency

- Circular dependencies are when two tables depend on each other
  - Foreign key reference each other
  - Programming version of what came first, chicken or the egg
- Example: A Student has be involved in exactly one Club, and a Club must be led by exactly one Student.




# Circular Dependency Example

A Student has be involved in exactly one Club, and a Club must be led by exactly one Student.

```
CREATE TABLE Student (  
  sid INTEGER PRIMARY KEY,  
  sname VARCHAR(100) NOT NULL,  
  club INTEGER NOT NULL,  
  FOREIGN KEY (club) REFERENCES Club  
);
```

```
CREATE TABLE Club (  
  cid INTEGER PRIMARY KEY,  
  cname VARCHAR(100) NOT NULL,  
  student_leader INTEGER NOT NULL,  
  FOREIGN KEY (student_leader)  
  REFERENCES Student  
);
```



Won't work!

Solution: first create both tables without dependencies, then add the dependencies after.

*set auto commit off*

```
insert into Student (cid);
```

```
insert into Club;
```

*commit.*

## Circular Dependency Example

A Student has be involved in exactly one Club, and a Club must be led by exactly one Student.

```
CREATE TABLE Student (  
  sid INTEGER PRIMARY KEY,  
  sname VARCHAR(100) NOT NULL,  
  club INTEGER NOT NULL  
);
```

```
CREATE TABLE Club (  
  cid INTEGER PRIMARY KEY,  
  cname VARCHAR(100) NOT NULL,  
  student_leader INTEGER NOT NULL  
);
```

```
ALTER TABLE Student ADD CONSTRAINT club_participation  
FOREIGN KEY (club) REFERENCES Club(cid) INITIALLY DEFERRED DEFERRABLE;
```

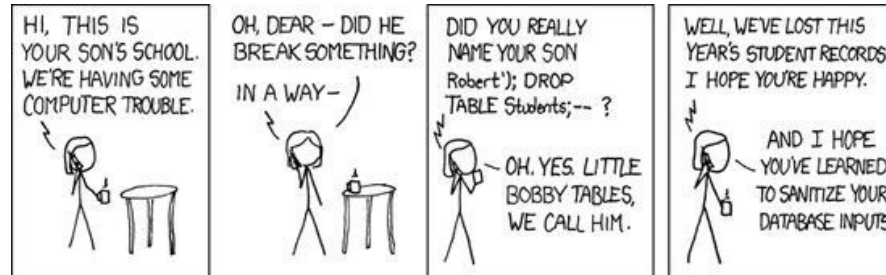
```
ALTER TABLE Club ADD CONSTRAINT leader_is_student  
FOREIGN KEY (student_leader) REFERENCES Student(sid) INITIALLY DEFERRED DEFERRABLE;
```

*⇒ constraint name.*

*defer constraints  
check at the  
end of transaction*

# Drop

- DROP TABLE Table\_Name CASCADE CONSTRAINTS;
- Deletes the table and all constraints it contains
  - includes foreign key constraints and triggers
  - does not include sequences





Get started with P1!