Review

ER Model -

- you should be able to look at and understand
- you should be able to draw one from a conceptual description

- 3 shapes to draw

Entities 

Relationships 

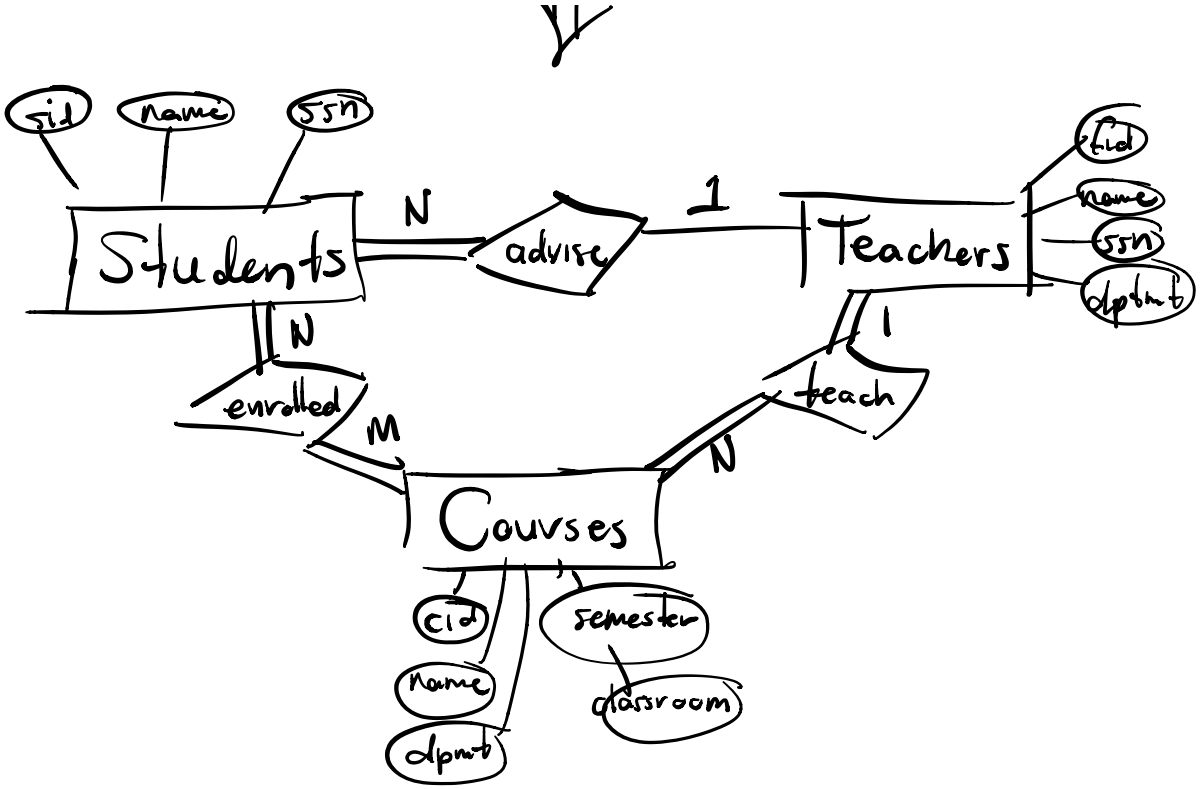
Attributes 

- the "arity" of relationships (1:1, 1:N, N:1, N:M)
  ↳ should be able to understand and label

- participation constraints

  ═══ (total) vs ——— (partial)

ex We want to draw an ER diagram for students, teachers, and the courses they participate in:



Relational Model -

- you should be able to convert from ER model to a relational model (and back)
- to express a relational model, just give the schema:

ex: student(sid, name, ssn, advisor_fid)
teachers(fid, name, ssn, dept)
courses(cid, name, dept, semester, classroom)
teach(fid, cid, how_long)
enrolled(sid, cid)

- you should be able to identify:

Superkeys - set of attributes that can uniquely identify a tuple in a relation

Candidate keys - a minimal superkey (a superkey where no subset is a superkey)

Primary keys - know that a primary key is some selected candidate key

- know that a "relation" is a set of tuples
- tuples ≈ rows
- relations ≈ tables
- attributes ≈ columns

Relational Algebra -

σ_p - Selection operator
- selects tuples that match predicate 'p'

π_{attrs} - Projection operator
- limits results to only the attributes in the 'attrs' list

$\times$ - Cartesian product operator
- match every tuple in one relation to every tuple in the other and results in a relation whose attributes are

the combined attributes of both participating relations

$\cup$ - Union operator
- results in all tuples from both relations
- can only be applied on relations w/ the exact same set of attributes

$-$ - difference operator
- $(R_1 - R_2)$ results in tuples in R_1 but not in R_2
- also only applies for relations w/ same attrs.

$\cap$ - Intersection operator
- results in tuples that exist in both relations
- also only applies for relations w/ same attrs.

$\bowtie_\theta$ - Theta join operator
- combination of $\times$ and σ_θ
- $R_1 \bowtie_\theta R_2 = \sigma_\theta (R_1 \times R_2)$

$\bowtie_{attrs}$ - Equijoin operator
- join on only the attributes in 'attrs'
- ex: $\text{Student} \bowtie_{sid} \text{Enrolled}$

$=$

$\text{Student} \bowtie_{s.sid = e.sid} \text{Enrolled}$

$\bowtie$ - Natural join
- join on all matching attributes between two relations
- ex. Student $\bowtie$ Teacher

$$\text{Student} \bowtie_{s.name = t.name \text{ and } s.ssn = t.ssn} \text{Teacher}$$

SQL Queries -

SELECT select_list
- same as π from relational algebra
- narrows down colums
- instead of select_list you can put * to get all columns

[DISTINCT]
- goes after SELECT to remove duplicate rows in the result

FROM from_list
- same as $\times$ operator between all tables in from_list
- know how to abbreviate table names
ex: (table1 t1, table2 t2 ... etc)

- know how to use those abbreviations elsewhere (def notation)
  ex: (t1.attr ... t2.attr)
- NOTE: remember that this only does cartesian product, so to perform a join, you need to include constraints in the WHERE clause

WHERE conditions
- same as the σ selection operator
- selects only rows that match 'conditions'
- you can build complex expressions in the WHERE clause
- ex: (and, &&, or, ||, =, !=, <, >, etc)
  (+, -, *, /, CONCAT(a,b))

<expr> AS <column_name>
- goes inside the SELECT clause and lets you use expressions to overwrite columns in the result, or create new columns

ex: (SELECT rating + 2 AS rating,
rating/2 AS half_rating...)

- again, the 'expr' can use any of the basic operators as with the WHERE clause

<string_column> LIKE <string-matching-expression>

- goes in the WHERE clause to do string matching beyond simple equality (=)
- we learned two matching characters for the matching expressions:
  - % - matches 0 or more arbitrary characters
  - _ - matches exactly 1 abitrary character

ex:

(WHERE name LIKE "B%")

- not case-sensitive

<query> UNION <query>
<table> <table>

- this goes between two full queries (or something resulting in a table) and gives the union (U) of the results.

ex:

(SELECT .... FROM ..... WHERE ....
UNION [ALL]
SELECT ... FROM .... WHERE ....)

[ALL] - if you want to retain duplicates

Example problems

Schema: sailors (sid, name, rating)
boats (bid, name, color)
reserv (sid, bid, date)

1) Find names of sailors who have reserved boat #2?

$\pi_{name}(\sigma_{bid=2}(\sigma_{s.sid=r.sid}(Sailors \times Reserv)))$

$\pi_{name}(\sigma_{bid=2}(Sailors \bowtie Reserv))$

$\pi_{name}(Sailors \bowtie_{s.sid=r.sid \wedge r.bid=2} Reserv)$

```
SELECT name
FROM sailors, reserv
WHERE sailors.sid = reserv.sid and reserv.bid = 2;
```

§

```
SELECT s.name
FROM sailors s, reserv r
WHERE s.sid = r.sid and r.bid = 2;
```

2) Find names of sailors who have reserved a blue boat?

$$\pi_{s.name}\left(\sigma_{b.color = \text{Blue}}\left(\sigma_{s.sid=r.sid \wedge b.bid=r.bid}\left(Sailors \times Reserv \times Boats\right)\right)\right)$$

$$\pi_{s.name}\left(\sigma_{b.color = \text{Blue}}\left(Sailors \bowtie_{sid} Reserv \bowtie_{bid} Boat\right)\right)$$

```
SELECT s.name
FROM Sailors s, reserv r, boats b
WHERE b.color = "Blue" AND s.sid = r.sid
    AND b.bid = r.bid;
```

---

3) Find colors of boats reserved by sailors named "Christie Weeks"?

$$\pi_{b.color}\left(\sigma_{s.name = \text{"Christie Weeks"}}\left(\sigma_{s.sid = r.sid \text{ AND } r.bid = b.bid}\left(Sailors \times Reserv \times Boats\right)\right)\right)$$

```
SELECT b.color
FROM sailors s, reserv r, boats b
WHERE s.name = "Christie Weeks" AND s.sid = r.sid AND
    r.bid = b.bid;
```

```
SELECT b.color
FROM sailors s INNER JOIN reserv r ON s.sid=
    INNER JOIN boats b ON r.bid=
```

WHERE s.sname = "Christie Weeks"; b.bid

---

4) Find the names of sailors who have reserved at least one boat?

$$\pi_{s.name}(\text{Reserv} \bowtie_{sid} \text{Sailors})$$

```
SELECT s.name
FROM reserv r, sailors s
WHERE r.sid = s.sid;
```

---

5) Find names of sailors who reserved <u>both</u> a red and a green boat?

$$\pi_{s.sname}\left[\left(\text{sailors} \bowtie_{sid} \text{reserv} \bowtie_{bid} \left(\sigma_{color=Red}(\text{boats})\right)\right) \cap \left(\text{sailors} \bowtie_{sid} \text{reserv} \bowtie_{bid} \left(\sigma_{color=green}(\text{boats})\right)\right)\right]$$

SQL? → we don't know how to do intersect ∩ yet
... so don't try