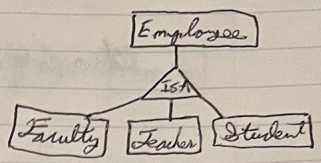
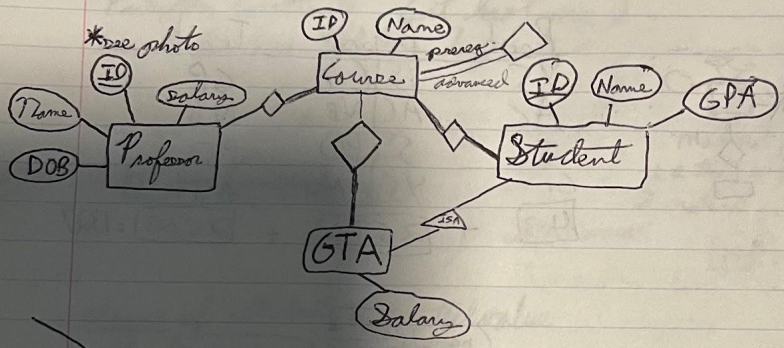
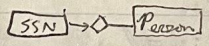
ISA Hierarchies A entity is considered a B entity also
points to parent entity
add descriptive attributes to subclass, identify entities that participate in relationships

person belongs only in one row - overlap
employees have to be hourly or contract - covering





HW1: 5/5 - Given a SSN, it only associates w/ one person. Person has any # of SSN.



assume any table does not have repeating entry

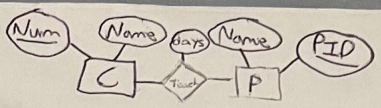
We have multiple unique identifiers - SSN ⇒ candidate key
- (SSN, name) ⇒ super key
subset of attribute is also a key

keys used to design DB → primary key, which is underlined

relation consists of relational schema - list of attribute/column names w/ data type
instance - tuples (rows, records)

schema specifies relation name, name of field and its domain, primary key underlined
- columns, attr,

relations are unordered, tuple is unique, demanded by relation definition



Teach (Num: number, PID: number) => Teach (cnum: number, PID: number, days: number)

number of tuples = cardinality (not row of attribute names)

degree = number of columns

→ bold arrow means one and only one

UniProf (salary, id, days, Name, PID) underline id or P.id

```
CREATE TABLE UniProf(
  salary REAL,
  id     char(10),
  days   INTEGER,
  Name   CHAR(10),
  PID    CHAR(10) UNIQUE,
  PRIMARY  (id)
);
```

* 1) write AS
2) translate to SQL
3) check candidate key
4) participation

→ not bold own(id, PID, days2) underline id or PID

```
CREATE TABLE UniAcct(
    salary REAL,
    id     CHAR(10),
  PRIMARY  KEY (id));
CREATE TABLE
Prof (
  Name   CHAR(10),
  PID    CHAR(10),
  PRIMARY  KEY(PID));   # entity set
```

```
*1) CREATE TABLE own (
2)      id  CHAR(10),
3)     PID  CHAR(10) UNIQUE,
4)    days  INTEGER,
      PRIMARY KEY(id),
5)    FOREIGN KEY(id)
        REFERENCE  UniAcct,
      FOREIGN KEY(PID)
        REFERENCE  Professor);
```

} Attr

reference UniAcct() table

5) participation

one bold arrow

CourseTeach(Name, CRN, PID) Professor(Name, PID)

```
CREATE TABLE CourseTeach(
Name CHAR(10),
CRN  CHAR(10),
PID  CHAR(10),  NOT NULL      # every course needs a prof.
PRIMARY KEY(CRN)
FOREIGN KEY(PID)
  REFERENCE Professor         # candidate key needs to be unique
```

example 4)

CourseTeach(Number, Name, PID)

```
CREATE TABLE CourseTeach (
      Number  CHAR(10),
      Name    CHAR(10),
      PID     CHAR(10), NOT NULL
      PRIMARY KEY(Number, PID)
      FOREIGN KEY(PID)
      REFERENCE Professor
      ON DELETE
        CASCADE );
```

example 5)

Manager(SSN, name, lot) Dept(did, dname, budget) Manages(SSN, Did, since)

```
CREATE TABLE Dept(
did  CHAR(10),
dname CHAR(10),
budget INTEGER
PRIMARY KEY(did))

CREATE TABLE Manager(
   ssn  CHAR(10),
   name  CHAR(10),
   lot   INTEGER,
   PRIMARY KEY(SSN))

CREATE TABLE Manages(
ssn     CHAR(10)
Did     CHAR(10)
since   DATE,
PRIMARY KEY(SSN, Did)
FOREIGN KEY(SSN)
  REFERENCE Manager
FOREIGN KEY(Did)
  REFERENCE Dept );
```

example 6) similar to ex5 but Manager (no more SSN PRIMARY KEY (DID))

NP × State Np.state = State.state

name	Np.state	State.state	capital
YS	WY	WY	Ch
~~YS~~	~~WY~~	~~TX~~	~~Au~~
~~BB~~	~~TX~~	~~WY~~	~~Ch~~
BB	TX	TX	Au

⋈ is easier

S.id of sailors who reserve both red & green boat?

```
(SELECT S.id
 FROM Reserve R, Boat B
 WHERE R.b.id = B.b.id
   AND B.color = 'red' )
INTERSECT
(SELECT Sid FROM Reserve R2, Boat B2 WHERE R2.bid = B2.bid
  AND B2.color = 'green')
```

Table A: sid of sailors who reserve red boat

B: sid of sailors ... green

two different modes

or green, all that changes is INTERSECT to UNION

red boat but never reserve green boat

change INTERSECT to EXCEPT

another solution

```
SELECT Rs.id Rname
FROM Reserve R, Boat B, Sailor S
     Reserve R2, Boat B2, Sailor S2
WHERE
   R.sid = R2.sid AND R.b.id = B.b.id AND R2.bid = B2.bid
   AND B.color = 'red' AND B2.color = 'green' AND
   B.b.id <> R2.b.id) AND S.s.id = R.sid AND S2.s.id = R2.sid
```

ER Model - entity relationship ≈ mind map

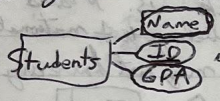
(object)

entity - object that exists & distinguishable from other obj., represented by set of attributes
concrete like person/book ; abstract like holiday/disease

entity set (entities w/ same type), not always disjoint (different entity sets, 1 entity)
all people have account at bank

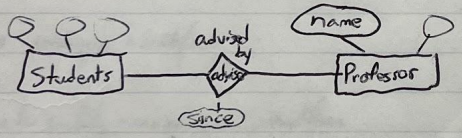
express logical structure graphically with ER diagram

entity set has 3 components: rectangles = name, ellipses = attributes, lines link attributes



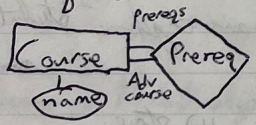
underline means it can uniquely define name

relationship : diamonds = relationship sets, lines linking ≥ 2 entity sets to relationship sets

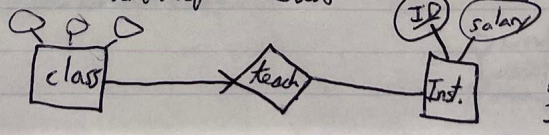


"a professor advises a student"

Prereqs for CSCI 4333 are CSCI 3333 & CMPE 3333



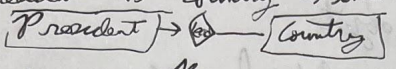
Key Constraints - number of entity to number of other entity
ex) student advised by only 1 professor, professor advises ≥ 1 student
- many-to-one constraint Bowling Club → was born in
- one-to-one constraint (1 man to 1 woman in marriage, or unmarried)
- one-to-many constraint (1 woman, 0 ≤ children, each child has only 1 mom)
mother of ← club
- many-to-many



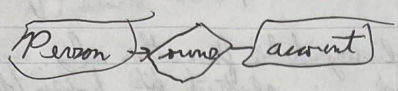
class can only be taught by 1 Inst.

my answers

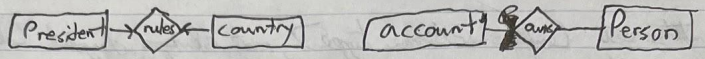
Current President vs Country → one to one constraint



Bank Account vs Person – many-to-many

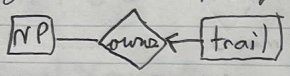


correct answers



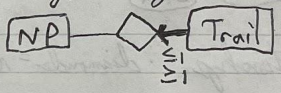
Participation Constraints (at least one)
cannot be described by key constraint

my answer: one to many



national park vs. trails
1. a trail only belong to 1 national park
2. national park may not have any trails ≥0

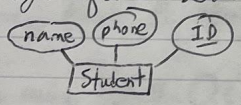
correct answer:



domain of attribute is permitted value
-9 for year not valid

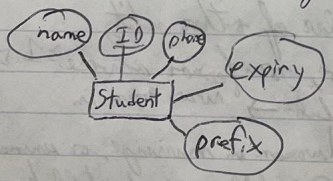
Dating App feet + inches better than just inches, millions of calculations needed

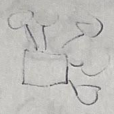
ER Diagram for Uni Database 1) each student registers name & phone 2) student has ID (associated)

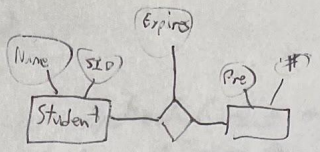


3) report expired date for phone 4) store pre-fix + number

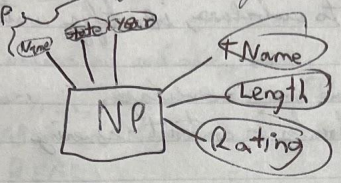
my answer:



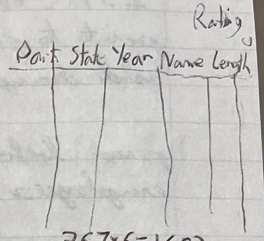




vs attribute of entity
entity set related to attribute by relationship set

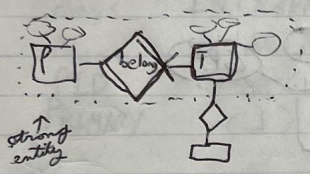


three entities



Trail

267×6=1602

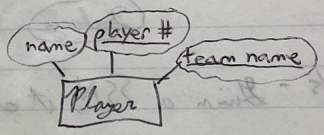


Part	Belong		Trail
YS	YS	GP...	GP
AC	AC	NB...	NB
S	S	PO...	PO
Gr	YS	MH	MH

4×3 + 26×2 + 26×3 = 1327

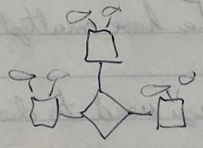
eliminate redundancy

key - unique identifier

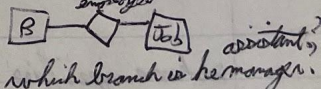


weak entity - no unique identifier
- associate with strong entity to identify
- ... is considered a weak entity

Ternary Relationships (higher order)

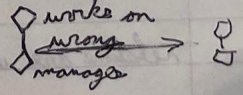


- can be converted into set of binary relationships
- employee at ≤ branch at job descriptions
- bad if subsets of entity ~~don't use that connection~~ need constraints



assistant?
which branch is he manager.

Aggregation
- manager see tasks of employee at another branch



example 6) similar to ex5 but Manager (no more SSN PRIMARY KEY (DId))

NP × State Np.state = State.state

name	Np.state	State.state	capital
YS	WY	WY	Ch
~~YS~~	~~WY~~	~~Tx~~	~~Au~~
~~BB~~	~~Tx~~	~~WY~~	~~Ch~~
BB	TX	TX	Au

⋈ is easier

S.id of sailors who reserve both red & green boat?

Table A: sid of sailors who reserve red boat

B: sid of sailors who green

```
(SELECT S.id
 FROM Reserve R, Boat B
 WHERE R.b.id = B.b.id
   AND B.color = 'red')
INTERSECT
(SELECT S.id FROM Reserve R2, Boat B2 WHERE R2.b.id = B2.b.id
 AND B2.color = 'green')
```

two different nodes
or green, all that changes is INTERSECT to UNION

red boat but never reserve green boat
change INTERSECT to EXCEPT

another solution

```
SELECT R.s.id Rname
FROM Reserve R, Boat B, Sailor S
     Reserve R2, Boat B2, Sailor S2
WHERE
    R.s.id = R2.s.id AND R.b.id = B.b.id AND R2.b.id = B2.b.id
    AND B.color = 'red' AND B2.color = 'green' AND
    B.b.id <> R2.b.id) AND S.s.id = R.s.id AND S2.s.id = R2.s.id
```

```
SELECT name FROM Sailor
WHERE rating<7 AND age>34
```

r>4, reserves all boats:

```
SELECT S.name FROM Sailor S
WHERE rating>4 AND
NOT EXISTS (
SELECT B.bid FROM Boat B
EXCEPT
SELECT B2.bid FROM Buy B2
WHERE B2.sid = S.sid
)
```

```
SELECT AVG(rating) FROM Sailor
```

```
SELECT company, MIN(rating) AS min_rating
FROM Sailor
GROUP BY company
```

for each yellow boat reserved by sailors from different companies, find total # of reservations

```
SELECT bid, COUNT(*) FROM
BUy B NATURAL JOIN Boat B2 NATURAL JOIN Sailor S
WHERE B2.color = "yellow"
GROUP BY bid
HAVING COUNT(DISTINCT S.company)>1
```

R(A,B,C) F = {A→C, C→A} fill according to F^+

A→C
C→A

B+ = B
C+ = AC
AB+ = ABC

~~AC+ = ABC~~ AC+=AC
BC+ = ABC
ABC+ = ABC

R(A,B,C,D,E) F = {A→DE, B→CD, A→B}

✓ A→DE, A→ABCDE
✓ A→B, A→ABCDE
✗ B→CD, B→ABCDE

decomposition: ABCDE

BCD | ABE BCNF

A	B	C
1	1	2
2	1	3
3	1	2
4	1	2
5	1	3

C→B ✓, every value is the same
B→A ✗

we need AC, AB decomposition
A is X, C is Y. X∩Y = AC

R is ABC, Y is C R−Y = ABC − C = AB