B561 MIDTERM REVIEW

Mo Zhou

What will be tested?

- similar to the **assignments**, as well as the **exercise questions** discussed in class

NOT included

- We will **NOT** test you on the following topics
  - DB application design
  - Interface design
  - Software engineering

What to take?

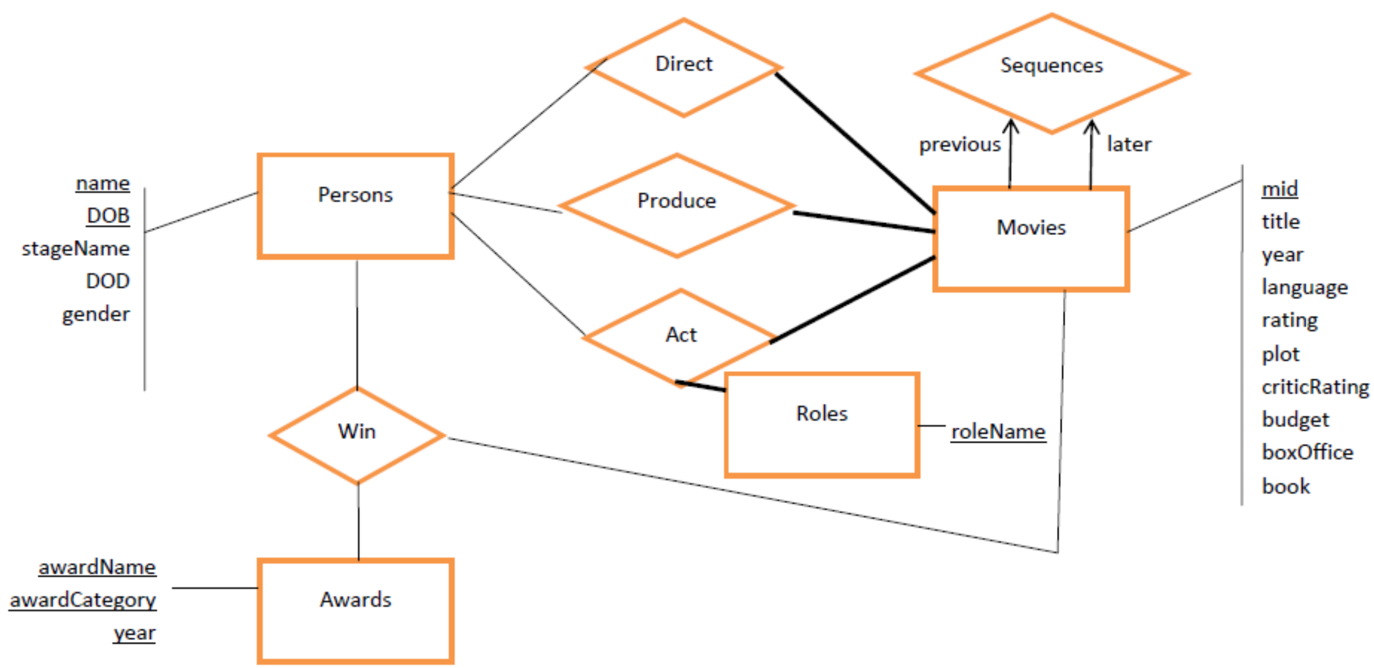
- Information sheet
- Pen, pencil

Database design

- Conceptual
  - ER model
- Logical
  - Schema
- Physical
  - Tables, index

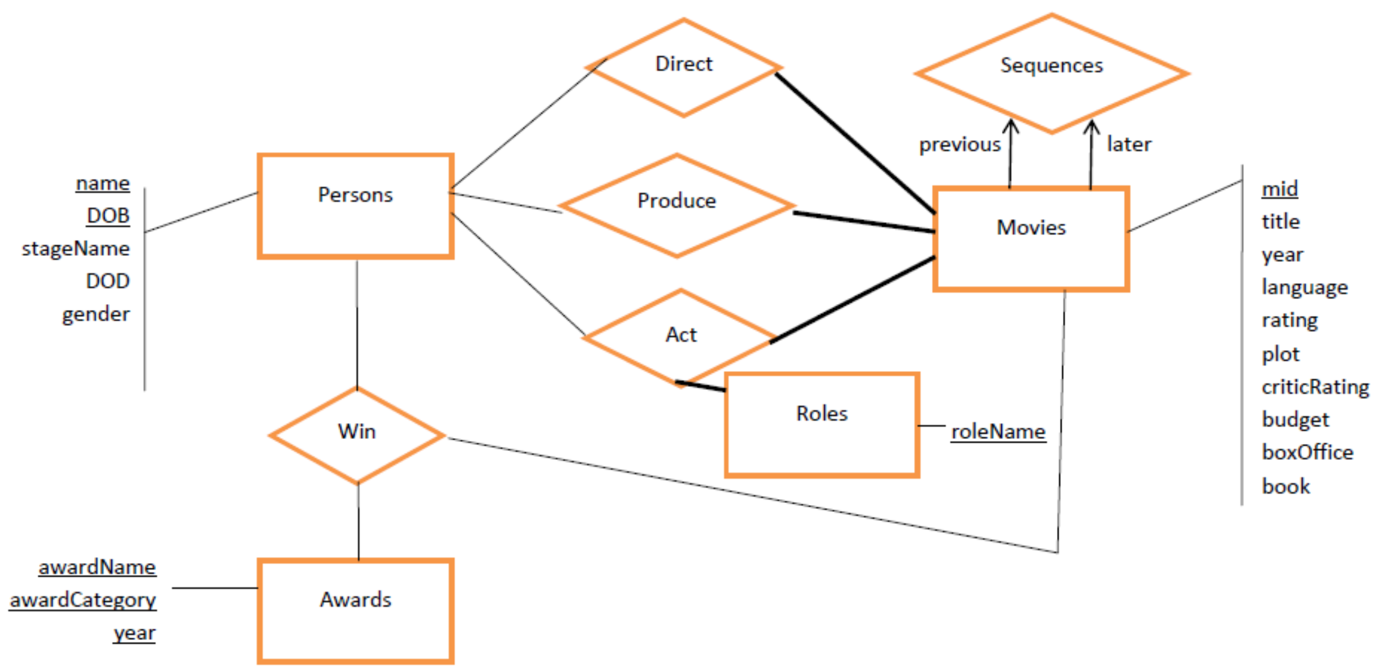
ER model

- Entity, attribute, relationship
- Binary vs n-nary relationship
- Key of entity and relationship



ER model

- Weak entity, class hierarchy
- Constraints: key, participation



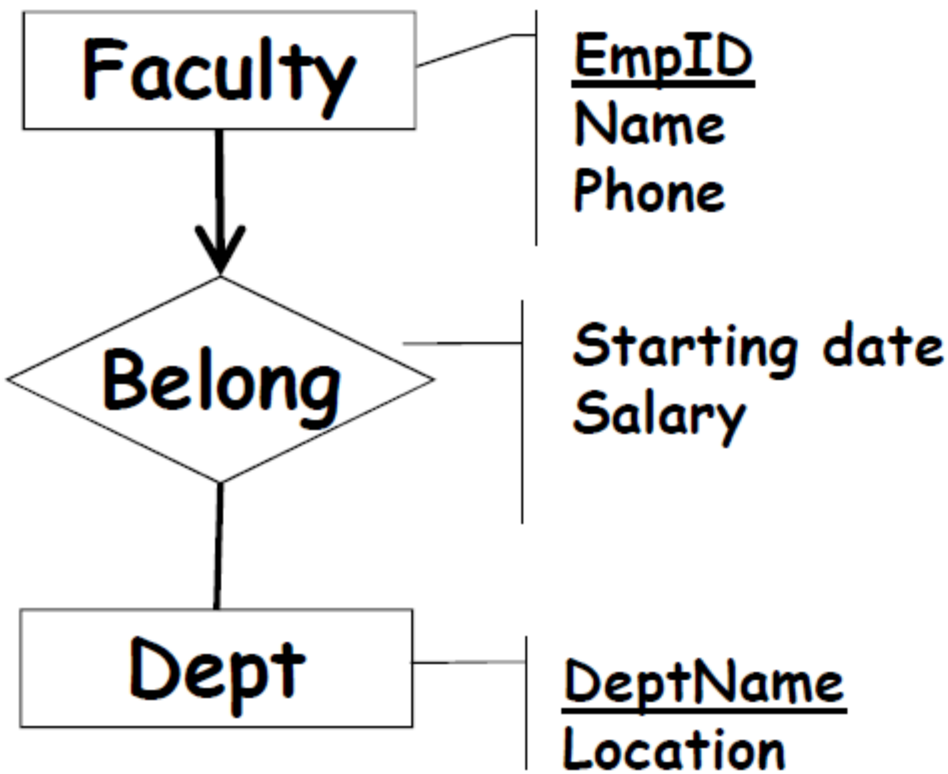
Relational model

- Relation, attribute, degree, cardinality
- Integrity constraints: domain, key, foreign key

Relational model

- ER → relational
  - Naïve transformation
  - Dealing with key constraint, participation constraints, weak entity, class hierarchy

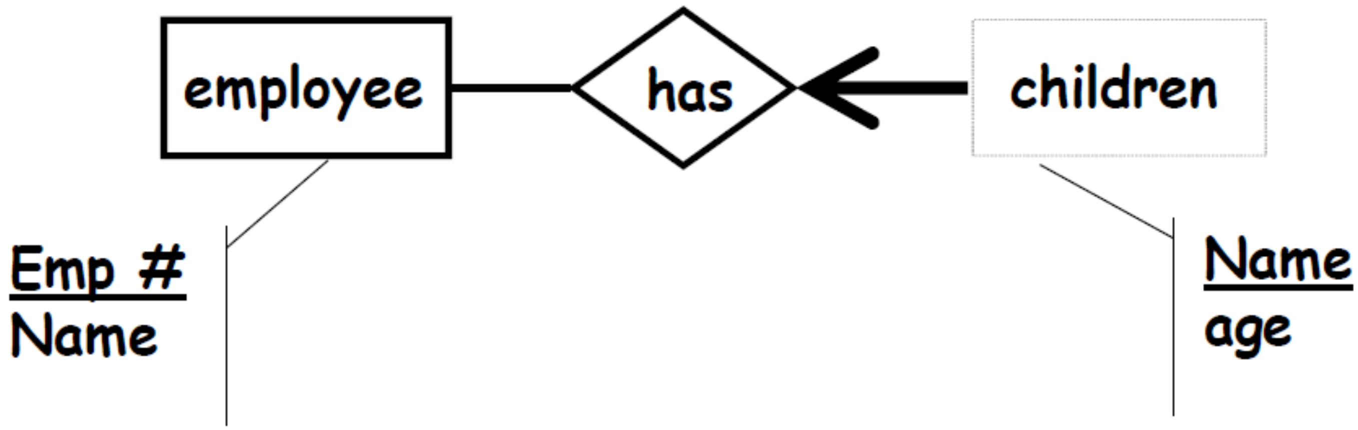
Example-key constraint




```
Create table Faculty_Belong(
EmpID int,
Name char(50),
Phone char(10),
DeptName char(50),
Start_data date,
Salary double,
Primary Key (EmpID)
Foreign Key (DeptName)
References Dept)

Create table Dept(
DeptName char(50),
Location char(500),
Primary Key (DeptName)
)
```

Example-weak entity



Create table Employee(
Emp int,
Name char(50),
Primary Key (Emp))

Create table hasChildren(
Emp int,
childName char(50),
childAge int,
Primary Key (Emp, Name)
Foreign Key (Emp) References Empoyee)

Functional dependency

- Functional dependency
  - In a given table, an attribute *Y* is said to have a functional dependency on a set of attributes *X* (written $X \rightarrow Y$) if and only if each *X* value is associated with precisely one *Y* value.
- Trivial dependency
  - A->A, AB->A
- Full dependency
  - AB->C but both A->C and B->C are not true
- Super key
- Candidate key

Functional dependency properties

- Reflexivity
  - If Y is a subset of X, then $X \rightarrow Y$
- Augmentation
  - If $X \rightarrow Y$, then $XZ \rightarrow YZ$
- Transitivity
  - If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- Union
  - If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- Decomposition
  - If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- Pseudotransitivity
  - If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$
- Set accumulation rule
  - If $X \rightarrow YZ$ and $Z \rightarrow AB$, then $X \rightarrow ZAB$

Closure, Equivalent, Cover

- Closure
  - Let F be a set of functional dependencies, the closer of F, F^+, is the set of all FDs that are implied by F.=
- Equivalent and Cover
  - Given a schema R and two functional dependencies F and G, if $F^+=G^+$ over R, $F \equiv G$. F is a cover of G
- Minimum cover
  - Given functional dependencies G, F is a minimum cover of G if
    - $F^+=G^+$
    - the right hand side of each FD in F is a single attribute
    - F is minimal, that is, if we remove any attribute from an FD in F or remove any FD from F, then F^+ will no longer equal G^+

Normal forms

- 1NF
  - No multi-value attributes, nested relation
- 2NF
  - A relation schema R is in second normal form (2NF) if every non-prime attribute A in R is fully functionally dependent on the primary key
- 3NF
  - R is in 2NF. Every non-prime attribute of R is non-transitively dependent (i.e. directly dependent) on every candidate key of R
- BCNF
  - R is in 3NF. Every nontrivial dependencies $X \rightarrow Y$, X is a superkey.

BCNF decomposition

- Let R be the initial table with FDs F
- S={R}
- **Until** all relation schemes in S are in BCNF
  for each R in S
  for each FD $X \rightarrow Y$ that violates BCNF for R
- $S = (S - \{R\}) \cup (R\text{-}Y) \cup (X,Y)$
- **enduntil**

- This is a simplified version. In words:
- When we find a table R with BCNF violation $X \rightarrow Y$ we:
  - 1] Remove R from S
  - 2] Add a table that has the same attributes as R except for Y
  - 3] Add a second table that contains the attributes in X and Y

Example

Let us consider the relation scheme R=(A,B,C,D,E) and the FDs:

$\{A\} \rightarrow \{B,E\}$, $\{C\} \rightarrow \{D\}$

Candidate key: AC

Step1

Pick $\{A\} \rightarrow \{B,E\}$

(A,B,C,D,E) generates R1=(<u>A,C</u>,D) and R2=(<u>A</u>,B,E)

Step2

Pick $\{C\} \rightarrow \{D\}$

{A, C, D} generates R1=(A,C) and R2=(C,D)

Final decomposition: R2=(A,B,E), R3=(A,C), R4=(C,D)

3NF decomposition

- Let R be the initial table with FDs F
- Compute the minimum cover Fc of F
- S=∅
- **for** each FD X→Y in the minimum cover Fc
  S=S∪(X,Y)
- **if** no scheme contains a candidate key for R
- Choose any candidate key CN
- S=S ∪ table with attributes of CN

Example

- R=(A, C, B, O)
- Functional dependencies (also the minimum cover):

  $\{B\} \rightarrow \{A, O\}$

  $\{C, A\} \rightarrow \{B\}$
- Candidate Keys: {C, A}
- $\{B\} \rightarrow \{O\}$ violates 3NF
- 3NF tables – for each FD in the minimum cover create a table
- R1= (<u>B</u>, A, O)
- R2= (<u>C, A</u>, B)

Queries

- Give a query, understand what it looks for
- Give data instance and query, compute the result
- Give schema and query requirement, write query using the language mentioned above

SQL

- Nested queries
- Group by, Having
- Aggregation
- Set operations
  - Attributes in both sets **MUST** be exactly the **SAME**

SQL

- Student (ID, name, address, age, GPA, SAT)
- Campus (location, enrollment, rank)
- Application (studentID, Univ, date, major, decision)

- **Find the youngest students**
- SELECT ID
- FROM Student
- WHERE age = (SELECT MIN(age) FROM Student);

SQL

- Student (ID, name, address, age, GPA, SAT)
- Campus (location, enrollment, rank)
- Application (studentID, Univ, date, major, decision)

- **Find the students whose applied universities overlap with the universities applied by 0001**
- SELECT DISTINCT ID, name
- FROM Student, Application
- WHERE ID=studentID and Univ in (
-     SELECT Univ
-     FROM Student, Application
-     WHERE ID = 0001 and ID=studentID);

SQL

- Student (ID, name, address, age, GPA, SAT)
- Campus (location, enrollment, rank)
- Application (studentID, Univ, date, major, decision)

- **Find the university that has more than 200 applicants whose GPAs are greater than 3.0.**
- SELECT.Univ
- FROM Student, Application
- WHERE ID=studentID and GPA > 3.0
- GROUP BY Univ
- HAVING Count(*)>200;

RA and RC

- If you are not good at writing equations directly, I suggest you write sql firstly.
- RA
  - SELECT clause➔ π
  - WHERE clause ➔ σ, $\bowtie$
  - EXCEPT ➔ -
  - UNION ➔ $\cup$
  - INTERSECT ➔ $\cap$

RA and RC

- RC
  - SELECT/FROM Clause
    - $\{p \mid \exists s \in R(s.attr1 = p.attr1 \wedge s.attr2 = p.attr2)\}$
    - R is the table where p is selected
  - WHERE clause
    - s.attr1>value
    - Joins relations R and T
      - $\exists s \in R, t \in T(s.attr1 = t.attr1)$

RA and RC

- Student (ID, name, address, age, GPA, SAT)
- Campus (location, enrollment, rank)
- Application (studentID, Univ, date, major, decision)

- **Find the ids and names of students with GPA higher than 3.7**
- SQL: **SELECT id, name** FROM Student **WHERE GPA>3.7**
- RA: $\pi_{id,\ name}(\sigma_{GPA>3.7}(Student))$
- RC: $p\ \exists s \in Student(s.ID = p.ID \wedge s.name = p.name \wedge s.GPA > 3.7)$

RA and RC

- Student (ID, name, address, age, GPA, SAT)
- Campus (location, enrollment, rank)
- Application (studentID, Univ, date, major, decision)

- **Find the name and GPA of the students whose GPA is higher than 3.7.**
- RA: $\pi\downarrow name\,(\sigma\downarrow GPA>3.7\ (Student))$
- RC: $p\exists s\in Student(s.name=p.name\wedge s.GPA>3.7\,)$

RA and RC

- Student (ID, name, address, age, GPA, SAT)
- Campus (location, enrollment, rank)
- Application (studentID, Univ, date, major, decision)

- **Find the names of the students who are younger than 20 and applied to IU.**
- SQL: **SELECT name** FROM Student, Application **WHERE ID=studentID and age<20 and Univ="IU";**
- RA: $\pi\downarrow \boldsymbol{name}\ (\boldsymbol{\sigma\downarrow age<20}\ (Student) \bowtie\downarrow \boldsymbol{ID=studentID}\ \boldsymbol{\sigma\downarrow Univ=IU}\ (Application))$
- RC: $\{p \mid \exists s \in Student, t \in Application$
- $(\boldsymbol{s.name=p.name \wedge s.age<20 \wedge s.ID=t.studentID \wedge t.Univ="IU"})$

RA and RC

- Student (ID, name, address, age, GPA, SAT)
- Campus (location, enrollment, rank)
- Application (studentID, Univ, date, major, decision)

- **Find the names and addresses of all students with GPA higher than 3.7 who applied to CS major at campus with enrollment less than 15,000 and were rejected.**
- RA: $\pi\downarrow name,address\ (\sigma\downarrow GPA>3.7\ (Student)\bowtie\downarrow ID=StudentID\ ((\ \sigma\downarrow enrollment<15000\ (Campus))$
- $\bowtie\downarrow location=Univ\ \ (\sigma\downarrow decision=reject\ and\ major="cs"\ (Application))))$
- RC
- $\{p|\ \exists s\in Student,c\in Campus,t\in Application(s.ID=t.studentID\wedge s.GPA>3.7\wedge t.major="cs"\wedge t.decision="reject"\wedge t.Univ=c.location\wedge c.enrollmentt<15000\wedge s.name=p.name\wedge s.address=p.address)\}$