Converting ER Diagrams to Tables

Gary KL Tam

Department of Computer Science
Swansea University

- An ER diagram is a pictorial representation of the information that can be captured by a database.
- Such a "picture" serves two purposes:
  - It allows database professionals to describe an overall design concisely yet accurately.
  - (Most of) it can be easily transformed into the relational schema.

DB Designs

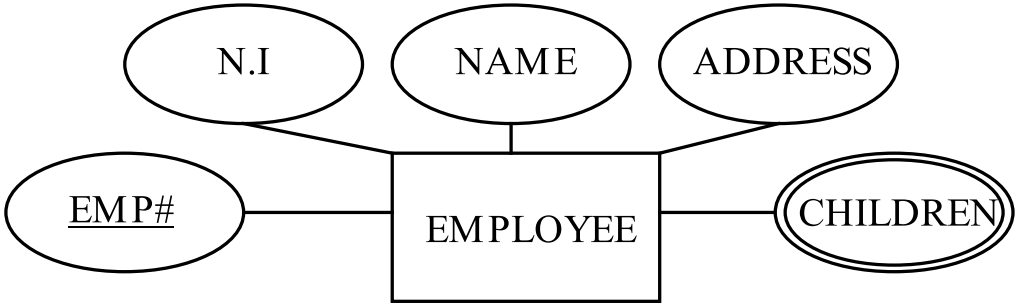
Given a ER digram (conceptual schema), it is not just a simple translation into a relational tables (relational schema).

- Not all ER constructs can be translated
- Reduce redundancy
- Improve efficiency

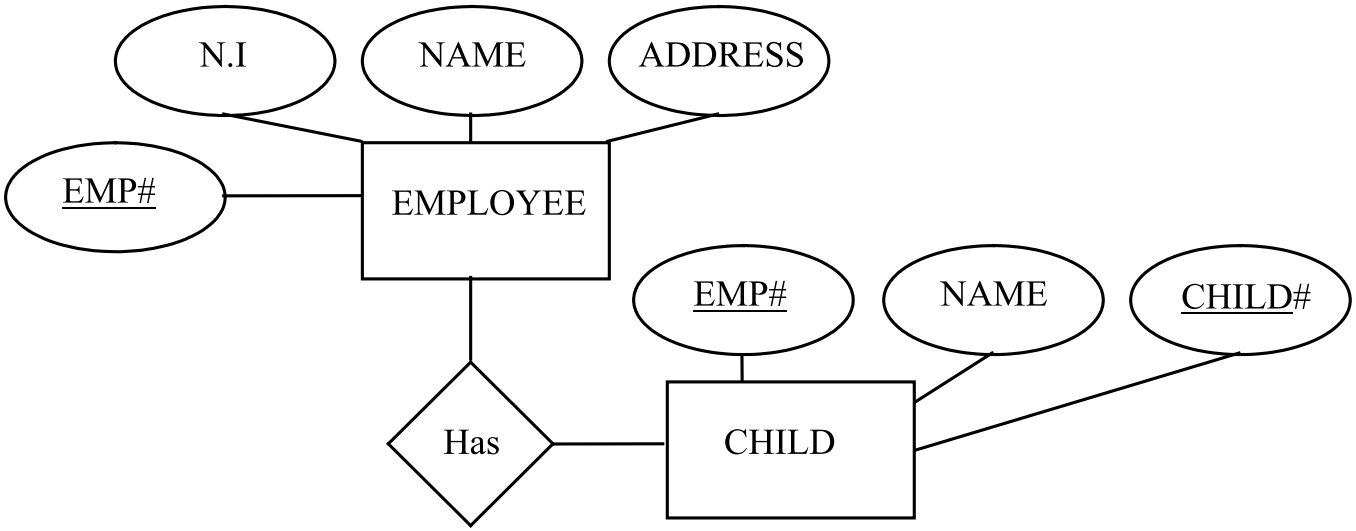
Steps:

- Restructure ER Diagrams
- Translation into relational schema

Multi-value attributes in a entity



- Ideally, multi-value attributes should be removed.
- Split into a relation and an weak entity



A Weak Entity

A weak entity depends on the existence of another entity

Example

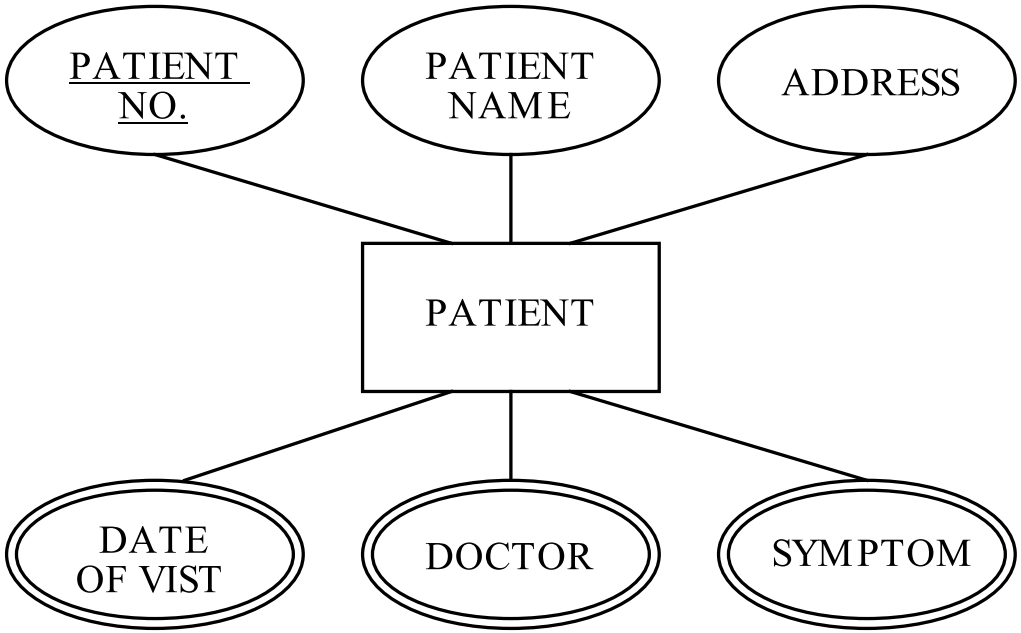
Child depends on the existence of a employee

- Weak entities often do not have a candidate key.
- But use primary key of parent entity
- Usually a result of:
  - Conscious choice by the data modeller
  - No global authority capable of creating a unique ID

Another Example

It is unlikely that there could be an agreement to assign unique player id across all football teams in the world.

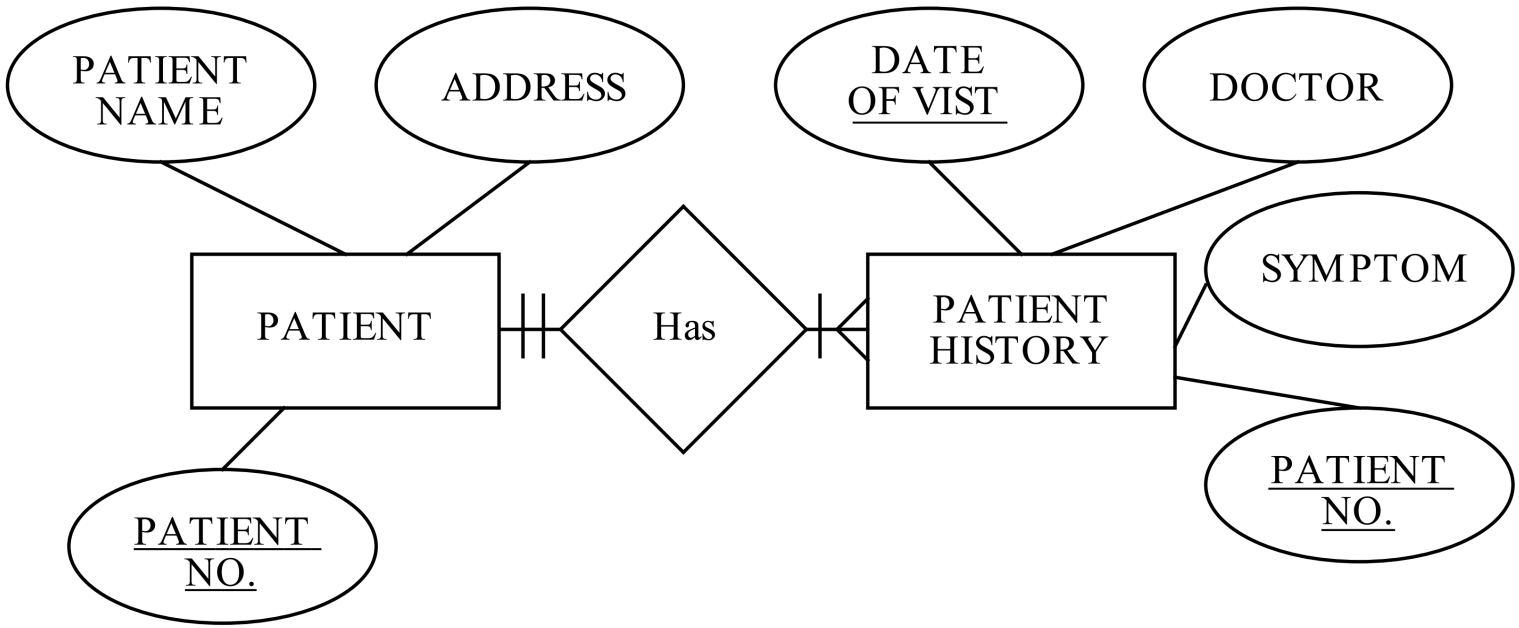
Repeating groups in a entity



- Look out: these multi-value attributes may be logically related.

Repeating groups in a entity

- For example, on each visit
  - Assumption 1: every patient see one doctor per day
  - Assumption 2: only see one doctor about one sympton
- This is only one example.
- We may create a doctor entity and make more relationships.



Entity Versus Attibutes

- A entity should satisfy at least one of the conditions:
  - it has at least one non-key attribute, or
  - it is the "many" in a many-one or many-many relationship
- Rules of thumb
  - A "thing" in its own right $\rightarrow$ entity (e.g. patient)
  - A "detail" about some other "thing" $\rightarrow$ attribute (e.g. name)
  - A "detail" correlated among many "things" $\rightarrow$ entity (e.g. patient history)
- Really this is just about avoiding redundancy

Challenge: modeling the real world

- Life is arbitrarily complex
  Directors who are also actors? Actors who play multiple roles in one movie? Animal actors?
- Design choices: Should a concept be modelled as an entity, an attribute, or a relationship?
- Limitations of the ER model: A lot of data semantics can be captured, but some cannot.
- Key to succesful model: parsimony
  - As complex as necessary, but no more
  - Choose to represent only "relevant" things

DB Designs

Given a ER digram (conceptual schema), it is not just a simple translation into a relational tables (relational schema).

- Not all ER constructs can be translated
- Reduce redundancy
- A improve efficiency

Steps:

- Restructure ER Diagrams
- Translation into relational schema

Translation into relational schema

Recall that an ER digram consists of:

- Entity
- Relationship

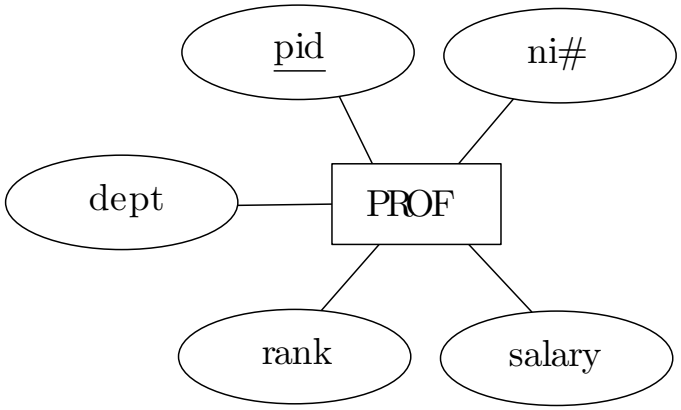
We will convert each entity/relationship to a table, which involves deciding the table's

- attributes
- candidate key
- and foreign key

Entity → Table

Given an entity set, create a table with the same attributes and candidate key.

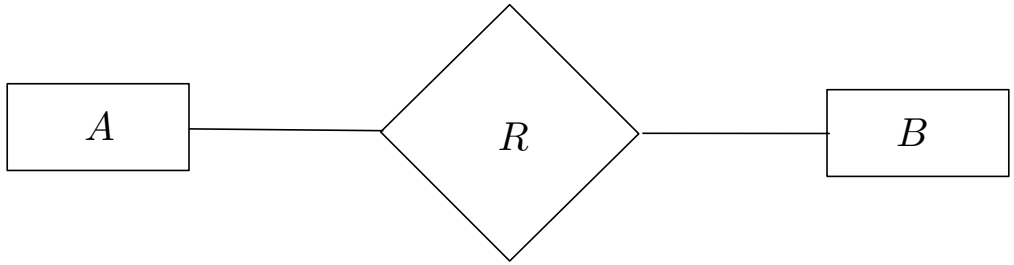
Example:



PROF (pid, ni#, dept, rank, salary)
where pid is set as a candidate key

Relationship → Table

Let us first consider binary relationship.
To convert R as shown below



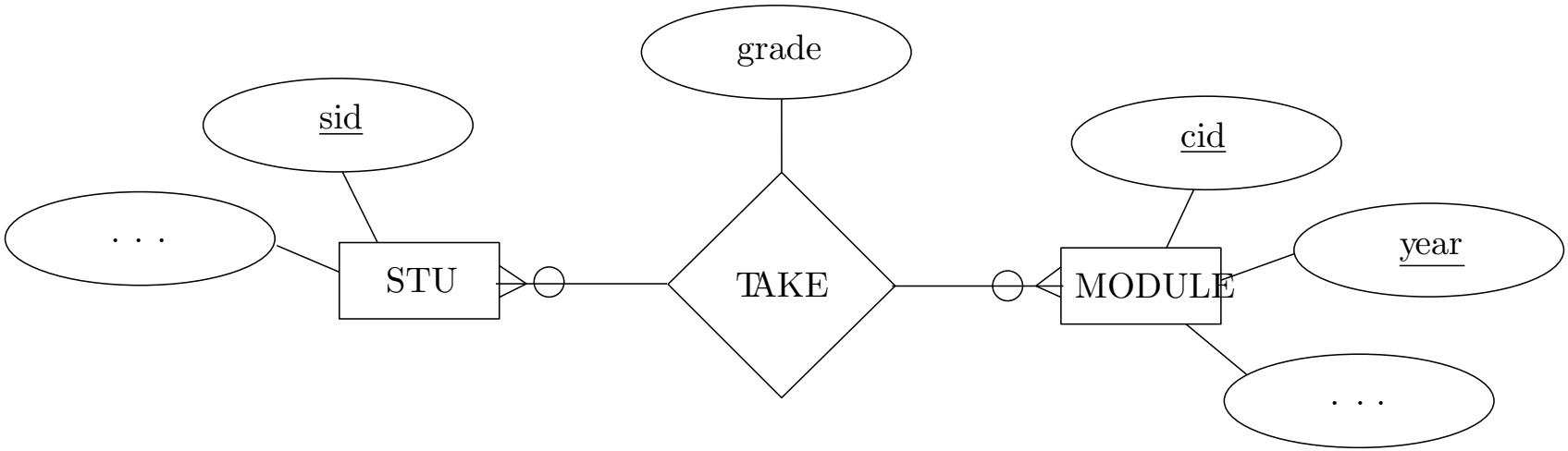
We create a table whose attributes include:

- the candidate keys of both A and B
  → foreign keys referencing A and B
- the attributes of R (if any).

The candidate key of the table, however, depends on the cardinality constraint of R. See the next few slides.

Relationship → Table

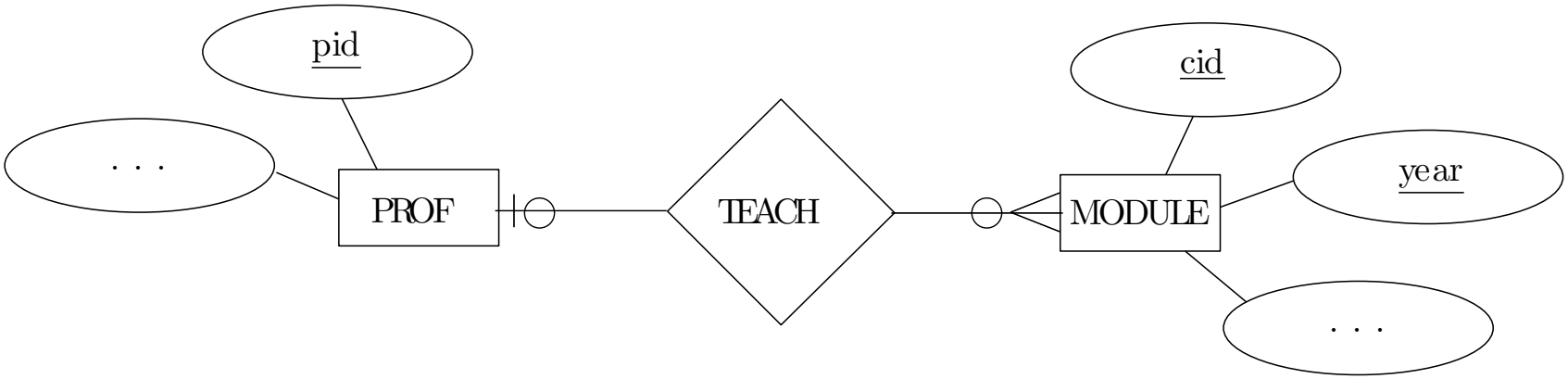
Many-to-many: The candidate key of R includes all the attributes in the candidate keys of A and B.
Example:



TAKE (sid, cid, year, grade)
where the candidate key is (sid, cid, year),
and two foreign key (sid) and (cid, year)

Relationship → Table

One-to-Many: The candidate key of R is the same as B (i.e., the entity set on the "many" side).
Example:



TEACH (pid, cid, year)
where the candidate key is (cid, year)

Think

Why not (pid, cid, year)?
What about if the module is not taught by any professor?

Think?

Why not (pid, cid, year)?

Note that every module can be taught by at most one professor. A module (of a particular cid, year) uniquely defines the professor! (cid, year) is the minimal, and so the candidate key. Example:
p1, c1, 2011
p2, c1, 2012

If module can be taught/shared by multiple professors?

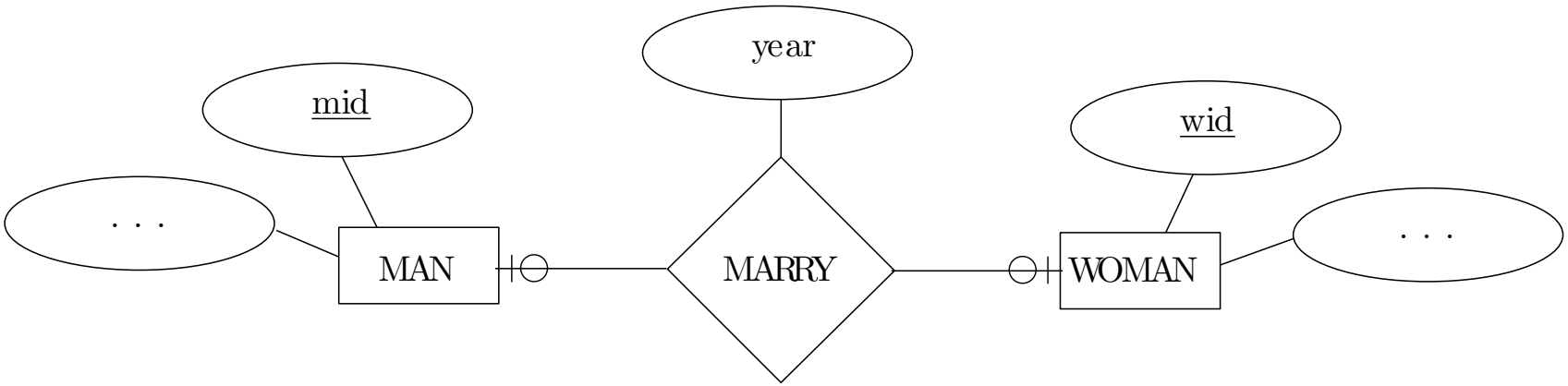
Then we need (pid, cid, year). Example:
p1, c1, 2011
p2, c1, 2011
p1, c1, 2012
p2, c1, 2012

What about if the module is not taught by any professor?

Then it is not in this TEACH table.

Relationship → Table

One-to-One: R has two candidate keys. The first (second) one is the same as A (B).
Example:



MARRY (mid, wid, year)
where the candidate key is mid, and another is wid.

Think

Why?

and two foreign keys (mid) and (wid)

Think?

Why not (mid, wid)?

On-to-one Relationship

Note that every mid/wid uniquely defines the relationship!

What about if mid not marry to any one?

then this relationship is not in the marry table.

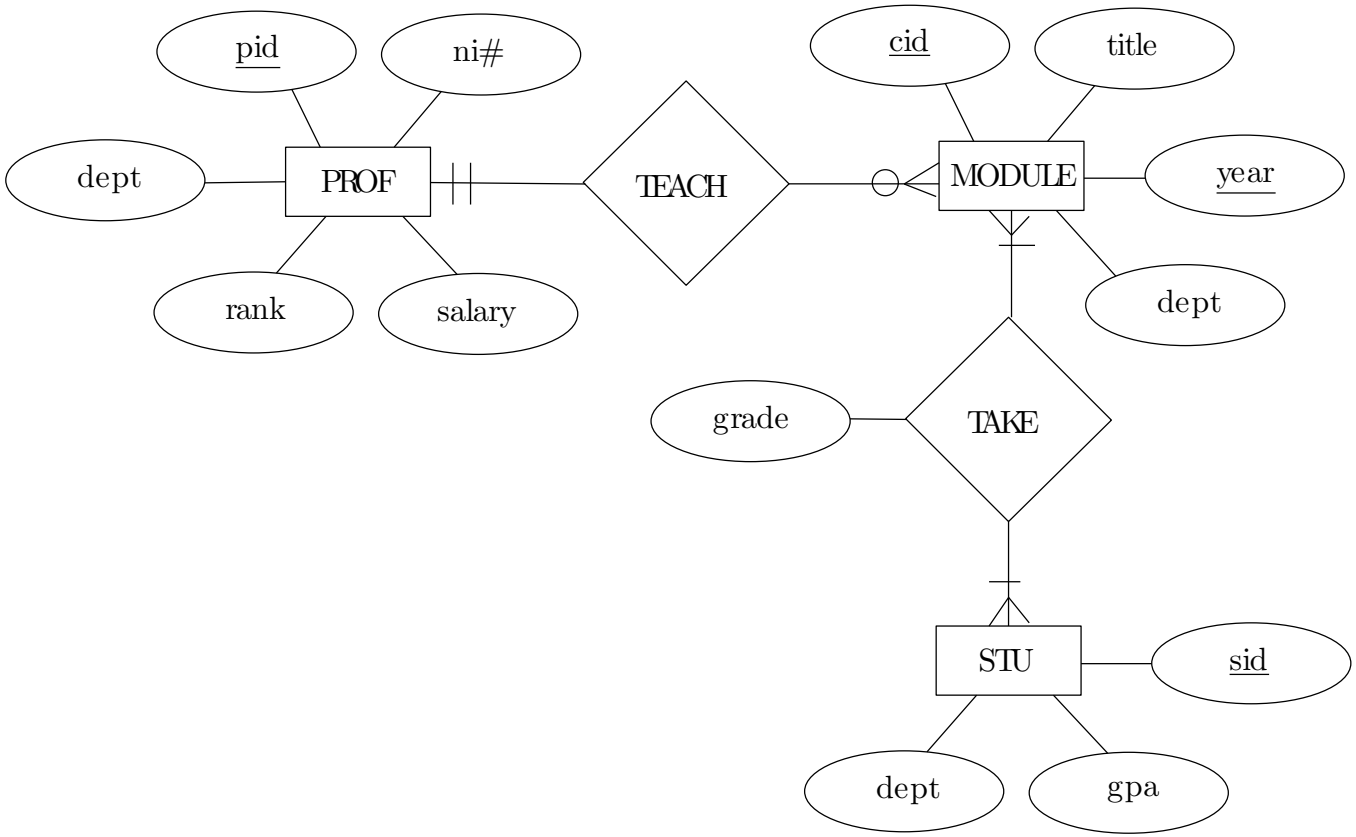
Relationship → Table

Multi-way relationship set R: Create a table that includes

- the candidate keys (and thus foreign keys) of the participating entity sets
- the attributes of R (if any).

The candidate key of the table includes all the attributes of the candidate keys of the participating entity sets.

Example



- PROF (pid, ni#, dept, rank, salary)
- MODULE (cid, year, title, dept)
- STU (sid, dept, gpa)
- TEACH (pid, cid, year), foreign key (pid), foreign key (cid, year)
- TAKE (cid, year, sid, grade), foreign key (sid), foreign key (cid, year)

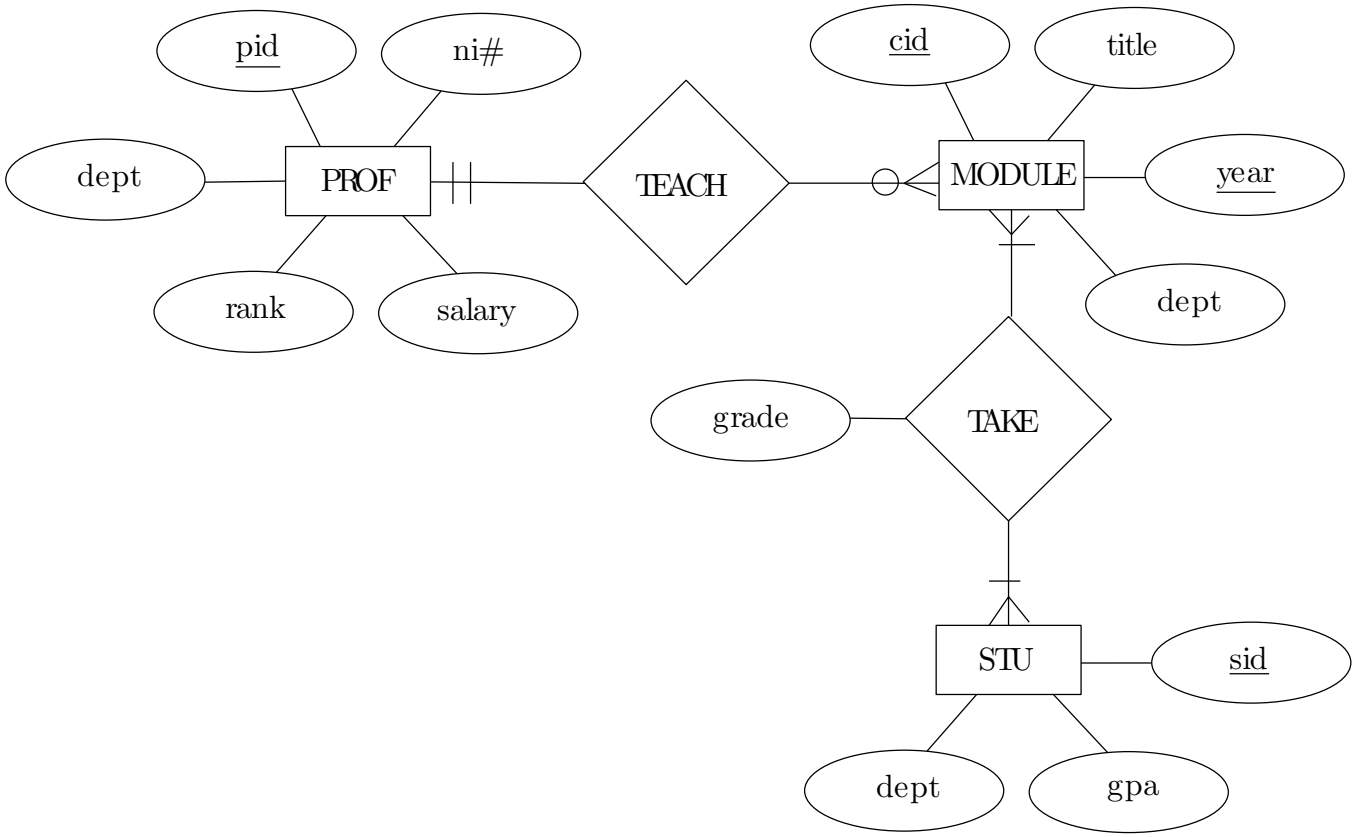
Example

What has not been captured by our conversion?

Participation constraints! For example, how to ensure that every class is taken by at least one student?

Participation constraints can be checked whenever necessary with SQL queries. Think: how?

Interestingly, the total participation of CLASS in the relationship set TEACH can be captured with a clever conversion. See the next slide.



- PROF (pid, ni#, dept, rank, salary)
- MODULE (cid, year, title, dept, pid), foreign key (pid)
- STU (sid, dept, gpa)
- TAKE (cid, year, sid, grade), foreign key (sid), foreign key (cid, year)

Note that there is no explicit table for TEACH.
Think: why doesn't the trick work for TAKE?

Total Participation:

- one-to... relationship with total participation on the 'one' side
- may not require relationship table
- enforced by foreign key in the entity

Partial Participation:

- Think:
  one-to.. relationship with partial participation on the 'one' side
- Can we do the same?

Yes. But for those tuple that does not participate, we will end up with NULL value in some tuple, which is not encouraged.

Summary

ER Model	Relational Model
Entity	Entity Table
1:1 or 1:N relationship	foreign key / relationship table
M:N relationship	relationship table and two foreign keys
n-ary relationship	relationship table with n foreign keys
simple attribute	attribute of table
multi-valued attribute	relation table and foreign key
key attribute	primary (or secondary key)

Will the schema be "good"?

- If we use this process, will the schema we get be a good one?
- The process should ensure that there is no redundancy
- But only with respect to what the ER diagram represents
- There are something the ER diagram do not consider: functional dependencies (We only have keys so far...)
- Our next two lectures will focus on normalization to further reduce redundancy.