Today …

• ER Modeling (cont)

Homework

• HW 3 due (today)
• HW 4 out (tomorrow)
• Quiz 4 on Thurs (normalization, ERDs)
**Weak Entity Sets**

In this model assume we ...  
(1). need to record the insurance policies of employees  
(2). need to track dependencies of policies  
(3). only need to store the name and date-of-birth of dependents  
(4). no longer track their policies or dependents when an employee leaves

We also assume dependents identified through their employees  
(5). we assume dependents of an employee have unique names  
(6). different employees could have dependents with the same name

In this case ...  
- Employee is the "**strong**" entity set  
- Policy is the "**identifying**" relationship set  
- Dependent is the "**weak**" entity set  
  - it wouldn't be in the DB if the strong entity were not present  
- Dependent name is a "**partial**" key  
  - must be combined with the strong entity key to identify the dependent
Weak Entity Sets in ERDs

- identifying relationship has double lines
- weak entity has double lines
- partial key has a dashed underline

Exercise 1: What can the cardinalities be for an identifying relationship?
Exercise 2: Are these two schemas are not equivalent?

- The **ternary** relationship implies that:
  
  *a supplier must be authorized to supply a part to a project*

- For example:
  
  - **office depot** supplies **pencils** to **project 112**
  - **staples** supplies **paper** to **project 115**
  - which **does not** imply **staples** supplies **pencils** to **112**
  - (but it would in the binary version ...)

![Diagram of Ternary vs Binary Relationships]
The binary relationships each represent something distinct

- For example, let's say that:
  - both staples and office depot provide pencils
  - office depot also provides paper
  - project 112 requires pencils
  - and project 112 authorizes both office depot and staples (as suppliers)

Q: Who supplies pencils to project 112?

- Unclear in the binary version ...
- providing, authorizing, and requiring to not imply supplying

Q: But is it possible to use binary relationships to capture supplying?

- Yes!
- Make supplies an entity set with three relationship sets
Dualities: Entities versus Attributes

Exercise 3: When should we model using entities instead of just attributes?

- If the attribute has attributes or other relationships
- Attributes are for “simple” (atomic) values
- Note: ER models do not have foreign keys!

Q: Should office be an attribute or an entity?

- It (again) depends on the application requirements!
- Many attributes can be “promoted” to an entity set
- Some entities can be “demoted” to an attribute value

Q: When would it make sense to model offices as entities?

- Employees can have more than one office
- Additional attributes (like area, location, etc.) of offices
- Offices are used in other relationships
  * e.g., to store the furniture, telephones, network drops in the offices

This is an example of why modeling can be hard ...

- and why it is important to understand the requirements
Translating ERDs to Relational Schemas

Entity sets

- Each entity becomes a separate table

Relationship Sets

- connect two (or sometimes more) entities
- we can either:
  - create a table for the relationship ... with entity keys as foreign keys
  - add a foreign key to an entity table

For many-to-many relationships

- create a new table to represent the relationship
- for example:

  TeamMember(ssn, num)

- with two foreign keys:

  Employee(ssn, name, lot)
  Team(num, name, start)

  TeamMember.ssn → Employee.ssn
  TeamMember.num → Team.num
For **one-to-many relationships**

- include a foreign key to the “many” side of the relationship
- for example:
  
  Department(`code`, `name`)
  
  Employee(`ssn`, `name`, `lot`, `homedept`)
  
  where Employee.homedept → Department.code

**Alternatively, for one-to-many relationships**

- create a new table (as in many-to-many)
- for example:
  
  HomeDepartment(`ssn`, `code`) ... note that ssn is the key!
  
  Department(`code`, `name`)
  
  Employee(`ssn`, `name`, `lot`)

**What are the trade-offs between these two approaches?**

- Joins
- In new table, store more information (extra ssn in HomeDepartment)
What if the relationship set has attributes?

- if many-to-many, store in the new relationship table
- if one-to-many, store in the table where relationship is represented

We'll talk about additional constraints later

- e.g., for 1..* cardinality constraints on many-to-many relationships
- minimum can be translated as an “inclusion” constraint (similar to an FK)

Exercise 4 & 5: ERD translation and trade-offs
Summary of Translation [Elmasri & Navathe]

1. Create table and choose key for each entity set (include its attributes)

2. Create table for each weak entity set, include attributes and the key of the owner as a foreign key. Set the key as foreign key plus partial key.

3. For each one-to-one relationship set, add a foreign key to one of the entity sets involved in the relationship (a foreign key to the other entity in the relationship). Foreign key should be made unique (also a key).

4. For each one-to-many relationship set, add a foreign key to the entity set on the many side of the relationship (to reference the entity set on the one side of the relationship).

5. For each many-to-many relationship set, create a new table. Include a foreign key for each participant entity set in the relationship set. The key for the new table is the set of all such foreign keys.

Note on Normalization ...

- If only binary relationships, things are easier ...
- Only needed if for an entity set there are non-trivial, non-key FDs
- Can then normalize these after translation as appropriate