Lecture 18:
• ER Diagrams (cont)

Announcements:
• Project Step 2 due Tues, Nov 8 (UI Design / Sketches)
• PS-3 due Tues, Nov 8
• Quiz 4 on Tues, Nov 8
• HW-5 out, due Thurs, Nov 10

Ternary (and n-ary) vs Binary Relationships

Check In: Are these two schemas equivalent?
• The ternary relationship implies that:
  a supplier must be authorized to supply a product to a project
Ternary (and n-ary) vs Binary Relationships

The ternary relationship implies that: *a supplier must be authorized to supply a product 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 project 112
- *but it would in the binary version ...*

Instead “promote” the ternary relationship set to an entity set

While nothing wrong with ternary (or higher) relationship sets ...  
- can be difficult to assign and interpret cardinality constraints  
- whereas using only binary relationships simplifies the constraints
When to use entities vs attributes?

- If has further attributes or relationships (e.g., office location)
- If entity can have many of them (e.g., many offices)
- Can depend on “importance” of item in domain

Similar issues with entities vs relationships (e.g., ternary to binary)

- special case: one-to-one relationships (again, a modeling decision)

Note: ER Diagrams never contain foreign keys! ... use relationships

Translating ERDs to Relations

(1) Entity sets
- Each entity becomes a separate table, with corresponding attributes
- Identifying attribute becomes primary key

(2) Relationship Sets: either ...
  (a) create a table for the relationship (with entities as foreign keys); or
  (b) add a foreign key to an entity table ... many-to-one, one-to-many
For **many-to-many relationships** ... must represent as a separate table

(i) Create a **new table** to represent the relationship ... TeamMember(ssn, num)

(ii) Make key composite of entity keys ... note can have additional attributes

(iii) Add foreign keys

  E.g.: TeamMember.ssn → Employee.ssn, TeamMember.num → Team.num

---

For **one-to-many / many-to-one relationships**

Option 1: Add a foreign key to the “many” side of the relationship

- Employee(ssn, name, lot, homedept)
- where Employee.homedept → Department.code
- note: homedept can be NULL here because of 0..1 constraint

Option 2: Create a separate table (like for many-to-many)

- HomeDept(ssn, code) ... why this key?
Translating ERDs to Relations (cont)

For Option 1 and 2:

Check In: What if the relationship set has attributes?

Check In: What are the trade-offs? (i.e., why choose one option over other)

Check In: How should weak entity sets be mapped?

Check In: What about normalization?

Final Note on Weak Entity Sets – Deletion Semantics

CREATE TABLE DependentPolicy (    name VARCHAR(20) NOT NULL,    ssn VARCHAR(11) NOT NULL,    dob DATE NOT NULL,    PRIMARY KEY (name, ssn),    FOREIGN KEY (ssn) REFERENCES Employee (ssn) ON DELETE CASCADE
);

Foreign key deletion constraints:
• ON DELETE RESTRICT disallows the deletion (the default)
• ON DELETE CASCADE parent row deletion causes child row deletion
• ON DELETE SET NULL sets child row’s FK to NULL