Logical Data Design

Using ER Methodology to Design Relational Database Schemas

The Development Process, 1

- Collect requirements. Analyze the requirements.
- Conceptually design the data (e.g., draw an ER diagram).
- Logically design the data (e.g., choose relation names and schemas).
- Physically design the data. Choose indices and other access structures.

The Development Process, 2

- Create the database. Insert sample data.
- Design and implement the application.
- Populate the database with “real” data.
- Gain experience running the application.
- Evaluate the earlier decisions and iterate, if necessary.

ER Diagrams: Primitives

- entity
- property
- relationship
- ISA

One-to-Many Relationships (“Key Constraints”)

Many-to-Many Relationships
Participation Constraints

Total

Partial

Relationships Have Instances

E1

E2

E1

E2

An ER Diagram, part 1

An ER Diagram, part 2

Ternary Relationships

Ternary Relationship Instances
Guidelines for ER Diagrams

- Be sure that entity classes correspond to collections.
- There should be no isolated entity classes, i.e., entity classes that aren't related to any other entity classes.
- Eliminate aggregate properties, set-valued properties, and foreign keys.
- Don't forget to underline properties that are part of a key.

ER Diagrams → Relations: Rule 1

- A strong entity class becomes a relation.
- The relation schema is identical to the entity class's set of properties.
- The key of the entity class becomes the primary key of the relation schema.

ER Diagrams → Relations: Rule 2

- A weak entity class becomes a relation.
- The relation schema contains the entity class's properties and the key of the entity class on which the weak entity class depends.
- The full key of the entity class becomes the primary key of the schema.
- The weak relationship “disappears.” If this relationship has any properties, they become attributes of the “weak” relation's schema.

ER Diagrams → Relations: Rule 3

- A many-to-many relationship becomes a relation.
- The relation schema contains the keys of the entity classes related by the relationship, plus any properties of the relationship.
- Each of the keys of the related entity classes is a foreign key of the new entity class.
- The union of the keys of the related entities becomes the primary key of the relation schema.

ER Diagrams → Relations: Rule 4

- A one-to-many relationship is “absorbed” by the relation corresponding to the entity class at the “many” side.
- This is done by adding to the schema of the relation corresponding to the “many” side the key of the relation corresponding to the “one” side.
- The added attributes are a foreign key.

ER Diagrams → Relations: Rule 4, cont.

- If the relationship has any properties, these are also added to the schema of the relation corresponding to the “many” side.
- If the one-to-many relationship connects a weak entity class with the entity class it depends on, this rule has no effect beyond the effect of Rule 2.
ER Diagrams → Relations: Rule 5

• An entity class E1 that is a subclass of an entity class E2 becomes a relation R1 whose schema includes E1’s properties and E2’s key.
• An instance of entity class E1 is represented by a tuple in R1 and a tuple in the relation (R2) corresponding to entity class E2.
• E2’s key becomes R1’s primary key.
• R1’s primary key is also a foreign key referring to R2.

Relation Schemas for Sample ER Diagram

• student(sid, surname, yob, gpa, sex, major)
• faculty(fid, fname)
• course(cid, name, description)
• course_off(cid, sec_no, semester, room, time_slot)

Relation Schemas for Sample ER Diagram, cont.

• enr(sid, cid, sec_no, semester, grade)
• prereq(prereq_of_cid, prereq_for_cid)

Revision:
• course_off(cid, sec_no, semester, room, time_slot, fid)  cid NOT NULL
• ta(sid, salary)

What About Participation Constraints?

• In the case of one-to-many relationships, a participation constraint (total participation) can be translated into a “NOT NULL” constraint on the foreign key:
• course_off(cid, sec_no, semester, room, time_slot, fid)  cid NOT NULL
• Constraints more complex and expensive to enforce in many-to-many case.

Conclusions

• ER methodology often yields a good relational schema design, because “real” objects get mapped onto database objects.
• Transformation to relation schemas can be automated. But choosing ER diagrams requires good judgment.
• ER methodology isn’t perfect. Other methodologies sometimes have to be applied.
• A good logical design sometimes yields a poor physical design (depending on the access structures supported by the database system). Design is an iterative process.