The Entity-Relationship Model

Excerpt from
Chapter 2, “Database Management Systems” 3ed, R. Ramakrishnan and J. Gehrke
Overview of Database Design

- Conceptual design: (ER Model is used at this stage.)
  - What are the entities and relationships in the enterprise?
  - What information about these entities and relationships should we store in the database?
  - What are the integrity constraints or business rules that hold?
  - A database `schema' in the ER Model can be represented pictorially (ER diagrams).
  - Can map an ER diagram into a relational schema.
**ER Model Basics**

- **Entity**: Real-world object distinguishable from other objects. An entity is described (in DB) using a set of attributes.

- **Entity Set**: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)
  - Each entity set has a key.
  - Each attribute has a domain.
**ER Model Basics (Contd.)**

- **Relationship**: Association among two or more entities. E.g., Henry works in Pharmacy department.
- **Relationship Set**: Collection of similar relationships.
  - An n-ary relationship set $R$ relates $n$ entity sets $E_1$ ... $E_n$; each relationship in $R$ involves entities $e_1 \in E_1$, ..., $e_n \in E_n$
    - Same entity set could participate in different relationship sets, or in different “roles” in same set.
Key Constraints

- Consider Works_In: An employee can work in many departments; a dept can have many employees.
- In contrast, each dept has at most one manager, according to the key constraint on Manages.
Participation Constraints

- Does every department have a manager?
  - If so, this is a participation constraint: the participation of Departments in Manages is said to be total (vs. partial).
    - Every did value in Departments table must appear in a row of the Manages table (with a non-null ssn value!)
Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.

![Diagram showing the relationship between Employees, Policy, and Dependents entities with attributes like ssn, name, lot, cost, pname, and age.]
Aggregation

- Used when we have to model a relationship involving (entity sets and) a relationship set.
  - **Aggregation** allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.

- **Aggregation vs. ternary relationship:**
  - Monitors is a distinct relationship, with a descriptive attribute.
  - Also, can say that each sponsorship is monitored by at most one employee.
Conceptual Design Using the ER Model

- **Design choices:**
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?

- **Constraints in the ER Model:**
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.
**Entity vs. Attribute**

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).
Entity vs. Attribute (Contd.)

- Works_In4 does not allow an employee to work in a department for two or more periods.

- Similar to the problem of wanting to record several addresses for an employee: We want to record several values of the descriptive attributes for each instance of this relationship. Accomplished by introducing new entity set, Duration.
Entity vs. Relationship

- First ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers all managed depts?
  - Redundancy: `dbudget` stored for each dept managed by manager.
  - Misleading: Suggests `dbudget` associated with department-mgr combination.
Binary vs. Ternary Relationships

- Constraints:
  - each policy is owned by just 1 employee
  - every policy must be owned by some employee
  - each dependent is tied to the covering policy

**Bad design**

**Better design**
Binary vs. Ternary Relationships (Contd.)

- Previous example illustrated a case when two binary relationships were better than one ternary relationship.
- An example in the other direction: a ternary relation **Contracts** relates entity sets **Parts**, **Departments** and **Suppliers**, and has descriptive attribute **qty**. No combination of binary relationships is an adequate substitute:
  - S “can-supply” P, D “needs” P, and D “deals-with” S does not imply that D has agreed to buy P from S.
  - How do we record **qty**?
Summary

- **Conceptual design follows requirements analysis,**
  - Yields a high-level description of data to be stored
- **ER model popular for conceptual design**
  - Constructs are expressive, close to the way people think about their applications.
  - Note: There are many variations on ER model.
- **Basic constructs:** *entities, relationships, and attributes* (of entities and relationships).
- **Some additional constructs:** *weak entities, aggregation.*
Several kinds of integrity constraints can be expressed in the ER model: key constraints, participation constraints. Some foreign key constraints are also implicit in the definition of a relationship set.

Some constraints (notably, functional dependencies) cannot be expressed in the ER model.

Constraints play an important role in determining the best database design for an enterprise.
Summary (Contd.)

- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, and whether or not to use aggregation.

- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.