The Entity-Relationship Model

Chapter 2, Chapter 3 (3.5 only)

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.
Overview of Database Design Example

- In an enterprise we want to keep track of the following facts:
  - The enterprise consists of several departments in which works several employees.
  - Each employee has a unique SSN and various information (name, DOB, address,..)
  - Each department has a unique DID and various information (name, budget,..)
  - A manager is an employee who currently manage only one department since a given date.
  - A department can have no more than one manager.

Overview of Database Design Example

- In an enterprise we want to keep track of the following facts:
  - Each employee works in only one department and some under the supervision of one other employee.
  - Employees either work by hour or work by contract.
  - Each employee has several dependents (with name, age) which can be enrolled for policy plans.
  - Each department sponsors several projects (with a unique PID, budget, start date) for a certain periods of time.
  - While a project is sponsored by a department it is monitored by an employee for till a certain date.
**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., Attishoo works in Pharmacy department.
- **Relationship Set**: Collection of similar relationships.
  - An n-ary relationship set \( R \) relates n entity sets \( E_1 \ldots E_n \); each relationship in \( R \) involves entities \( e_1 \ldots E_1, \ldots, e_n \ldots E_n \)
  - 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 Departments entity must appear in an instance of the Manages relationship.
**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.

**ISA (`is a’) Hierarchies**

- As in C++, or other PLs, attributes are inherited.
- If we declare A *ISA* B, every A entity is also considered to be a B entity.
  - *Overlap constraints:* Can Joe be an Hourly_Emps as well as a Contract_Emps entity? *(Allowed/disallowed)*
  - *Covering constraints:* Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify entities that participate in a relationship.
**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**

- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.
- What are the additional constraints in the 2nd diagram?
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 \( qty \)?

Logical DB Design: ER to Relational

- Entity sets to tables:

```sql
CREATE TABLE Employees
(ssn CHAR(11),
name CHAR(20),
lot INTEGER,
PRIMARY KEY (ssn))
```
**Relationship Sets to Tables**

- In translating a relationship set to a relation, attributes of the relation must include:
  - Keys for each participating entity set (as foreign keys).
  - This set of attributes forms a superkey for the relation.
  - All descriptive attributes.

```sql
CREATE TABLE Works_In(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
    REFERENCES Employees,
  FOREIGN KEY (did)
    REFERENCES Departments)
```

**Review: Key Constraints**

- Each dept has at most one manager, according to the *key constraint* on Manages.

```
1-to-1  1-to Many  Many-to-1  Many-to-Many
```

Translation to relational model?
Translating ER Diagrams with Key Constraints

- Map relationship to a table:
  - Note that did is the key now!
  - Separate tables for Employees and Departments.

- Since each department has a unique manager, we could instead combine Manages and Departments.

```sql
CREATE TABLE Manages(
    ssn CHAR(11),
    did INTEGER,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```

```sql
CREATE TABLE Dept_Mgr(
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    ssn CHAR(11),
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees)
```

Review: 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!)
Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to CHECK constraints).

```sql
CREATE TABLE Dept_Mgr(
did INTEGER,
dname CHAR(20),
budget REAL,
ssn CHAR(11) NOT NULL,
since DATE,
PRIMARY KEY (did),
FOREIGN KEY (ssn) REFERENCES Employees,
ON DELETE NO ACTION)
```

Review: 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 (1 owner, many weak entities).
  - Weak entity set must have total participation in this identifying relationship set.
Translating Weak Entity Sets

- Weak entity set and identifying relationship set are translated into a single table.
  - When the owner entity is deleted, all owned weak entities must also be deleted.

```sql
CREATE TABLE Dep_Policy (  
  pname CHAR(20),  
  age INTEGER,  
  cost REAL,  
  ssn CHAR(11) NOT NULL,  
  PRIMARY KEY (pname, ssn),  
  FOREIGN KEY (ssn) REFERENCES Employees,  
  ON DELETE CASCADE)
```

Review: ISA Hierarchies

- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.

- Overlap constraints: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (Allowed/disallowed)
- Covering constraints: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (Yes/no)
Translating ISA Hierarchies to Relations

- **General approach:**
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
    - *Hourly_Emps:* Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (*hourly_wages, hours_worked, ssn*); must delete Hourly_Emps tuple if referenced Employees tuple is deleted.
    - Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.
  - **Alternative:** Just Hourly_Emps and Contract_Emps.
    - *Hourly_Emps:* *ssn, name, lot, hourly_wages, hours_worked.*
    - Each employee must be in one of these two subclasses.

---

Review: Binary vs. Ternary Relationships

- **What are the additional constraints in the 2nd diagram?**

**Bad design**

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

- The key constraints allow us to combine Purchaser with Policies and Beneficiary with Dependents.
- Participation constraints lead to NOT NULL constraints.
- What if Policies is a weak entity set?

```sql
CREATE TABLE Policies (  
  policyid INTEGER,  
  cost REAL,  
  ssn CHAR(11) NOT NULL,  
  PRIMARY KEY (policyid),  
  FOREIGN KEY (ssn) REFERENCES Employees,  
  ON DELETE CASCADE)
```

```sql
CREATE TABLE Dependents (  
  pname CHAR(20),  
  age INTEGER,  
  policyid INTEGER,  
  PRIMARY KEY (pname, policyid),  
  FOREIGN KEY (policyid) REFERENCES Policies,  
  ON DELETE CASCADE)
```

---

**Summary of Conceptual Design**

- **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.
- **Basic constructs:** entities, relationships, and attributes (of entities and relationships).
- **Some additional constructs:** weak entities, ISA hierarchies, and aggregation.
- **Note:** There are many variations on ER model.
Summary of ER (Contd.)

Several kinds of integrity constraints can be expressed in the ER model: *key constraints, participation constraints, and overlap/covering constraints* for ISA hierarchies. 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 of ER (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, whether or not to use ISA hierarchies, 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.
- Rules to translate ER to relational model.