Data Modeling using the Entity-Relationship (ER) Model

"The bad news is we have to amputate your liver. The good news is it'll be great with onions!"
Overview of Database Design

❖ “Conceptual Schema” design:
  ▪ What are the Entities and Relationships (ER) of the enterprise?
  ▪ What information about these entities and relationships should be stored in the database?
  ▪ What are the integrity constraints or rules that hold?
  ▪ A database “model” can be represented pictorially (ER diagrams), but they are seldom used in practice.
  ▪ ER models are often used to construct relational database.
Other Data Models

❖ Hierarchical
   - Tree-based
   - Data is partitioned into smaller and smaller groups to facilitate searching and enumeration

❖ Network
   - Graph-based
   - Datatypes are “linked” to other datatypes
   - Hierarchical and relational are specializations of network models

❖ Object-Oriented
   - Adds inheritance to the Network model to allow for new, related datatypes
ER Modeling

- **Entity**: A thing distinguishable from other things. Entities are characterized by a set of attributes.

- **Entity Set**: A collection of 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 one or more *key* attributes that uniquely identifies it. By convention, the key is indicated by underlining.
  - Each attribute has a *domain*. (a *type* with possible constraints)

<table>
<thead>
<tr>
<th>ENTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute$_1$</td>
</tr>
<tr>
<td>Attribute$_2$</td>
</tr>
<tr>
<td>Attribute$_3$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
</tr>
<tr>
<td>FirstName</td>
</tr>
<tr>
<td>LastName</td>
</tr>
<tr>
<td>JobTitle</td>
</tr>
</tbody>
</table>
**Relationship**: Associations between Entities. e.g., David works in the Math 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 \( \{(e_1', \ldots, e_n') \mid e_1' \in E_1', \ldots, e_n' \in E_n'\} \), a.k.a a tuple
- Same entity set could participate in different relationship sets (a member of multiple departments), or in different “roles” in same set (a manager is also an employee).
Key Constraints

- Consider Works_In: An employee can work in many departments; a dept can have many employees.
- In contrast, each dept might have only one manager, placing constraints on Reports_To.
Participation Constraints

❖ Must 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, which relates each department to the employee who manages it.
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**

- It is often useful to partition entities into classes, like in an OOL.
- If we declare A **ISA** B, every A entity is also considered to be a B entity.

- **Overlap constraints**: Can Joe be an Salary_Emp as well as a Hourly_Emp entity? *(Allowed/disallowed)*
- **Covering constraints**: Does *every* Employee entity also have to be either an Salary_Emp or a Hourly_Emp entity? *(Yes/no)*
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify entities that participate in a relationship.

---

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>FirstName</td>
<td>LastName</td>
<td>JobTitle</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SALARY_EMP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Salary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HOURLY_EMP</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>HrsWorked</td>
<td>HourlyWage</td>
</tr>
</tbody>
</table>
Aggregation

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

Aggregation vs. ternary or higher order relationships:
- Monitors is a distinct relationship with its own descriptive attributes.
- Allows constraints on attribute subsets. A project is monitored by 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 models.
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 themselves be sets (multivalued)).
  • If the structure (city, street, etc.) is important, e.g., we want to retrieve employees from a given city, *address* must be modeled as an entity (since attribute values are atomic).
Entity vs. Attribute (Contd.)

- Works_In does not allow an employee to work in a department for two or more periods, or track historical information.

- 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 set 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 2\textsuperscript{nd} 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.
  ▪ Where do we record qty?
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.
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.
Next Time

- Setup an environment
- Look at files
- Basic file “model”
- Think about how scan and process data