Introduction to Data Management

Lecture #3
(Conceptual DB Design)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

- Reminders:
  - Sign up on Piazza! (About 10% still haven’t done this...)
  - First real quiz in discussions next week! (Study! 😊)
- HW #1 is now available
  - New startup company: “TopicalBirds.com”
  - First step: Develop an E-R model for them!
  - A word on timing/logistics/changes...
- Today’s topic(s): Conceptual DB design
  - Basic E-R concepts
  - E-R schema design
  - Advanced E-R features
- Any lingering Q’s from last time?
Overview of Database Design

- **Conceptual design**: (ER Model 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 (using an ER diagram).
  - Can map an ER diagram into a relational schema (manually or using a design tool’s automation).

ER Model Basics

- **Entity**: Real-world object, distinguishable from all other objects. An entity is described (in DB-land) 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 get to ISA hierarchies…)
  - Each entity set has a key (a unique identifier); this can be either one attribute (an “atomic” key) or several attributes (called a “composite” key)
  - Each attribute has a domain (similar to a data type).
ER Model Basics (Contd.)

- **Relationship**: Association among two or more entities. E.g., Santa Claus works in the Toy 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 \( e_1:E_1, \ldots, e_n:E_n \)
  - One entity set can participate in different relationship sets – or in different “roles” in the same set.

Cardinality 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 **cardinality constraint** on Manages above.

(Note: A given employee can manage several departments)
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 below must appear in an instance of the Manages relationship
  - Ditto for both Employees and Departments for Works_In

ER Basics: Another Example

- Let’s see if you can read/interpret the ER diagram above...! (©)
  - What attributes are unique (i.e., identify their associated entity instances)?
  - What are the rules about (the much coveted) parking passes?
  - What are the rules (constraints) about professors being in departments?
  - And, what are the rules about professors heading departments?
Another Example (Answers)

- **Unique attributes:**
  - Professor.fac_id, Dept.dno, Parking Space.pid

- **Faculty parking:**
  - 1 space/faculty, one faculty/space
  - Some faculty can bike or walk (.cycle)
  - Some parking spaces may be unused

- **Faculty in departments:**
  - Faculty may have appointments in multiple departments
  - Departments can have multiple faculty in them
  - No empty departments, and no unaffiliated faculty

- **Department management:**
  - One head per department (exactly)
  - Not all faculty are department heads

**NOTE:** These things are all “rules of the universe” that are just being modeled here!

**Q:** Can a faculty member head a department that he or she isn’t actually in?

Another Example (E’s & R’s)

```
S1 S2 S3
P1 P2 P3 P4
D1 D2 D3
```

- **Parking Spaces**
  - Assigned (1:1)

- **Professors**
  - In (M:N)

- **Departments**
  - Head (1:N)
**Weak Entities**

- A *weak entity* can be identified uniquely only by considering the primary key of some other (*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.
  - Dependent identifier is unique only *within* owner context (__), so its fully qualified key here is (ssn, dname)

![Image of weak entity diagram]

**Ternary Relationships (and beyond)**

- A prescription is a 3-way relationship between a patient, a doctor, and a drug; with the cardinality constraints above:
  - A given patient+drug will be associated with *one* doctor (1)
  - A given patient+doctor may be associated with *several* drugs (N)
  - A given doctor+drug may be associated with *several* patients (M)

**General note:** Relationship key ≤ (entity keys)
ISA (“is a”) Hierarchies

- As in Java or other PLs, ER attributes are inherited (including the key attribute).
- If we declare A ISA B, every A entity is also considered to be a B entity.
  - Covering constraints: Must every Employees entity be either an Hourly_Emps or a Contract_Emps entity? (Yes or no)
    - Ex: Hourly_Emps AND Contract_Emps COVER Employees (pick 1 of 2 vs. 1 of 3)
  - Overlap constraints: Can some Employees entity be an Hourly_Emps as well as a Contract_Emps entity? (Allowed or disallowed)
    - Ex: Hourly_Emps OVERLAPS Contract_Emps (else pick 1 of the 3 types)
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify subclasses that participate in a relationship.
- Design: specialization (top-down), generalization (bottom-up)

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 participating in (other) relationships.

Aggregation vs. ternary relationship:

- Monitors is a distinct relationship; even has its own attribute here.
- Each sponsorship can monitored by zero or more employees (as above).
**Additional Advanced ER Features**

- Multi-valued (vs. single-valued) attributes
  - Employees
    - ssn
    - name
    - phone

- Derived (vs. base/stored) attributes
  - Employees
    - ssn
    - name
    - bdate
    - age

- Composite (vs. atomic) attributes
  - Employees
    - ssn
    - name
    - address
      - street
      - city
      - zip
    - snum

*NOTE:* Can model (two of) these using additional entity and relationship types.