Conceptual Design. The Entity-Relationship (ER) Model

Conceptual Design – ER Model

- What are the entities and relationships in a typical application?
  - What information about these entities and relationships should we store in the database?

- What are the integrity constraints or business rules
  - Key constraints
  - Participation constraints

- Representation through ER diagrams
  - ER diagrams are then mapped into relational schemas
  - Conversion is fairly mechanical

Entities and Entity Sets

- **Entity**: represents a real-world object
  - Characterized using set of attributes
  - Each attribute has a domain – similar to variable types

- **Entity Set**: represents collection of similar entities
  - E.g., all employees in an organization
  - All entities in an entity set share same set of attributes

Keys

- Each entity set has a key
  - Set of attributes that uniquely identify an entity
  - Multiple candidate keys may exist
  - Primary key selected among them

Entity Set Representation

Representation Convention:
- Entity sets: rectangles
- Attributes: ovals, with key attributes underlined
- Edges connect entity sets to attributes
Relationships and Relationship Sets

- **Relationship**: Association among two (or more) entities
  - “Gabriel works in CS department”
  - Can have descriptive attributes: e.g., “since 9/1/2011”
  - But relationship must be fully determined by entities!
  - Binary, ternary or multi-way (n-way) relationships

- **Relationship Set**: Collection of similar relationships
  - Contains n-tuples (e₁, …, eₙ), where eᵢ belongs to entity set Eᵢ
  - Instance: “snapshot” of relationship set at some point in time

Visualizing Relationships and Rel. Sets

- **Edge** = Relationship
- Set of Edges = Relationship Set

Relationship Set Representation

- Employees → Works_In → Departments

Representation Convention:
- Relationship sets: diamonds
- Edges connect relationship sets to entity sets, and relationship sets to relationship set attributes

A Special Case of Relationship

- An entity set can participate in a relationship set with itself
- Entities in same set play different roles in the relationship
- **Role indicators** express the role

Example 1

- **Works_In relationship**: an employee can work in many departments; a dept can have many employees.
  - *many-to-many*
Example 2

- **Manages** relationship: each dept has **at most one manager**
  - one-to-many
  - from Employees to Departments
  - or
  - many-to-one
  - from Departments to Employees

Participation Constraints

- **Total vs Partial Participation**
  - **Total**: every department must have a manager
    - “Departments” entity set has total participation in relationship
    - Represented as thickened line (there is a key constraint as well)
  - **Partial**: not every employee is a manager
    - “Employees” entity set has partial participation

Example

Design a database for a bank, including information about customers and their accounts. Information about customers includes their name, address, phone and SSN. Accounts have numbers, types (e.g., savings/checking) and balances.

1. Draw the E/R diagram for this database.
2. Modify the E/R diagram such that each customer must have at least one account.
3. Modify the E/R diagram further such that an account can have at most one customer.

Mapping ER to Relational Schemas

- For most part, process is mechanical
  - Some special cases arise in the presence of constraints

- Translation from ER to SQL requires:
  - Mapping entity sets to tables
  - Mapping relationship sets to tables
  - Capturing key constraints
  - Capturing participation constraints

Entity Sets to Tables

CREATE TABLE Employees
(ssn CHAR(11),
nname CHAR(20),
lot INTEGER,
PRIMARY KEY (ssn))
Relationship Sets to Tables

- "No-constraints" case follows simple rules
- Relationship set becomes a relation, attributes include:
  - Keys for each participating entity set (as foreign keys pointing to respective entity table)
  - All descriptive attributes for relationship
  - Primary key of relationship set table is the concatenation of primary keys for the entity sets

What if there are Key Constraints?

- Each department has at most one manager, according to the key constraint on Manages

Variant 1

- Map relationship to a table:
  - Note that did is the key now!
  - Separate table for Manages relationship.

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)

Variant 2

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

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

Review: Participation Constraints

- Does every department have a manager?
  - If yes, the participation of Departments in Manages is total
  - Every did value in Departments table must appear in a row of the Manages table (with a non-null ssn value!), but this cannot be controlled in SQL (unless we use complex constraints)
  - Turns out that it is NOT possible to capture this with the two-tables mapping
    - Foreign key mechanism does not allow to check if there is a reference to every tuple in the referenced table
    - The Dept_Mgr variant is the only way!
Participation Constraints in SQL

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

Participation Constraints Summary
- General case
  - Total participation cannot be enforced unless we use complex constraints
- What if there is also a key constraint in place?
  - If the entity set with total participation also has a key constraint, then it is possible to capture total participation
  - But only if “combined” table construction is used!

Design Choices in the ER Model
- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?
  - Considers hierarchies and inheritance
  - Outside the scope of this class

Entity vs. Attribute
- Should address be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Sometimes address may have to be an entity:
  - If we have several addresses per employee (since attributes cannot be set-valued)
  - If the structure (city, street, etc.) is important, e.g., retrieve employees in a given city (attribute values are atomic!)