CSE 532 – Theory of Database Systems
Lecture 04 (Chapter 4)
Conceptual Modeling of Databases with E-R Model and UML

Ilchul Yoon
Assistant Professor
State University of New York, Korea

Database Design

- Goal: specification of database schema
- Methodology
  - Use E-R model to get a high-level graphical view of essential components of enterprise and how they are related
  - Convert E-R diagram to DDL – use rules

- E-R Model: enterprise is viewed as a set of
  - Entities
  - Relationships among entities
Entities

- **Entity (instance):** an object involved in the enterprise
  - Ex: John, CSE532
- **Entity Type:** set of similar objects
  - Ex: students, courses
- **Attribute:** describes one aspect of an entity type
  - Ex: name, maximum enrollment

Entity Type

- **Entity type described by set of attributes**
  - Person: Id, Name, Address, Hobbies
- **Domain:** possible values of an attribute
  - Value can be a set (in contrast to relational model)
    - (111111, John, 123 Main St, {stamps, coins})
- **Key:** minimum set of attributes that uniquely identifies an entity (candidate key)
- **Entity Schema:** entity type name, attributes (and associated domain), key constraints
Entity Type (con’t)

- Graphical Representation in E-R diagram:

Relationships

- Relationship: relates two or more entities
  - John majors in Computer Science
- Relationship Type: set of similar relationships
  - Student (entity type) related to Department (entity type) by MajorsIn (relationship type).
- Distinction:
  - relation (relational model) - set of tuples
  - relationship (E-R Model) – describes relationship between entities of an enterprise
  - Both entity types and relationship types (E-R model) may be represented as relations (in the relational model)
Attributes and Roles

- **Attribute** of a relationship type describes the relationship
  - e.g., John majors in CS since 2000
    - John and CS are related
    - 2000 describes relationship - value of SINCE attribute of MajorsIn relationship type

- **Role** of a relationship type names one of the related entities
  - e.g., John is value of Student role, CS value of Department role of MajorsIn relationship type
  - (John, CS; 2000) describes a relationship

Relationship Type

- Described by set of attributes and roles
  - e.g., MajorsIn: Student, Department, Since
  - Here we have used as the role name (Student) the name of the entity type (Student) of the participant in the relationship, but ...
Roles

- Problem: relationship can relate elements of same entity type
  - e.g., ReportsTo relationship type relates two elements of Employee entity type:
    - Bob reports to Mary since 2000
    - We do not have distinct names for the roles
    - It is not clear who reports to whom

Roles (con’t)

- Solution: role name of relationship type need not be same as name of entity type from which participants are drawn
  - ReportsTo has roles Subordinate and Supervisor and attribute Since
  - Values of Subordinate and Supervisor both drawn from entity type Employee
Schema of a Relationship Type

- Role names, $R_i$, and their corresponding entity sets
  - Roles must be single valued
  - number of roles = degree of relationship
- Attribute names, $A_j$, and their corresponding domains
  - Attributes may be set valued
- Key: Minimum set of roles and attributes that uniquely identify a relationship

- Relationship: $<e_1, ... e_n; a_1, ... a_k>$
  - $e_i$ is an entity, a value from $R_i$’s entity set
  - $a_j$ is a set of attribute values with elements from domain of $A_j$

Graphical Representation

- Roles are edges labeled with role names
  - Omitted if role name = name of entity set
  - Most attributes have been omitted.
Single-role Key Constraint

- If, for a particular participant entity type, each entity (instance) participates in at most one relationship (instance), corresponding role is a key of relationship type
  - E.g., Professor role is unique in WorksIn

Representation in E-R diagram: arrow

Cardinality Constraints

- Defined on role and restricts the number of relationship instances in which an instance of connected entity type can participate.
Cardinality Constraints and Single-Role Key Constraints

- Two ways of representation

![Diagram](image1)

Many-to-One, One-to-One, and Many-to-Many Correspondence
Entity Type Hierarchies

- One entity type might be subtype of another
  - Freshman is a subtype of Student
- A relationship exists between a Freshman entity and the corresponding Student entity
  - e.g., Freshman John is related to Student John
- This relationship is called IsA
  - Freshman IsA Student
  - The two entities related by IsA are always descriptions of the same real-world object

IsA

![IsA Diagram]

- Represents 4 relationship types

```
IsA
  /   \
/     \
Freshman Sophmore Junior Senior
```
Properties of IsA

- **Inheritance** - Attributes of supertype apply to subtype.
  - E.g., GPA attribute of Student applies to Freshman
  - Subtype inherits all attributes of supertype.
  - Key of supertype is key of subtype

- **Transitivity** - Hierarchy of IsA
  - Student is subtype of Person, Freshman is subtype of Student, so Freshman is also a subtype of Person

Advantages of IsA

- Can create a more concise and readable E-R diagram
  - Attributes common to different entity sets need not be repeated
  - They can be grouped in one place as attributes of supertype
  - Attributes of (sibling) subtypes can be different
IsA Hierarchy - Example

Constraints on Type Hierarchies

- Might have associated constraints:
  - **Covering** constraint: Union of subtype entities is equal to set of supertype entities
    - Employee is either a secretary or a technician (or both)
  - **Disjointness** constraint: Sets of subtype entities are disjoint from one another
    - Freshman, Sophomore, Junior, Senior are disjoint set
Participation Constraint

- If every entity participates in at least one relationship, a participation constraint holds:
  - A participation constraint of entity type E having role ρ in relationship type R states that for e in E there is an r in R such that ρ(r) = e.
  - e.g., every professor works in at least one department

Participation and Key Constraint

- If every entity participates in exactly one relationship, both a participation and a key constraint hold:
  - e.g., every professor works in exactly one department
Representing Entity Types in Relational Model

- An entity type corresponds to a relation
- Relation’s attributes = entity type’s attributes
  - Problem: entity type can have set valued attributes, e.g.,
  - Person: Id, Name, Address, Hobbies
  - Solution: Use several rows to represent a single entity
    - (111111, John, 123 Main St, stamps)
    - (111111, John, 123 Main St, coins)
  - Problems with this solution:
    - Redundancy
    - Key of entity type (Id) not key of relation
    - Hence, the resulting relation must be further transformed (Chapter 6)

Representing Relationship Types in Relational Model

- Typically, a relationship becomes a relation in the relational model
- Attributes of the corresponding relation are
  - Attributes of relationship type
  - For each role, the primary key of the entity type associated with that role
Representing Relationship Types in Relational Model (cont’d)

- **Example:**

  ![Diagram showing the relationship between Courses, Teaching, and Professor tables]

  - Courses (CrsCode, SectNo, Enroll)
  - Professor (Id, DeptId, Name)
  - Teaching (CrsCode, SectNo, Id, RoomNo, TAs)

  ![Not in the key]

Representing Relationship Types in Relational Model (cont’d)

- **Candidate key of corresponding table = candidate key of relation**
  - Except when there are set valued attributes
  - **Example:** Teaching (CrsCode, SectNo, Id, RoomNo, TAs)
    - Key of relationship type = (CrsCode, SectNo)
    - Key of relation = (CrsCode, SectNo, TAs)

  ![Set valued]

<table>
<thead>
<tr>
<th>CrsCode</th>
<th>SectNo</th>
<th>Id</th>
<th>RoomNo</th>
<th>TAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSE305</td>
<td>1</td>
<td>1234</td>
<td>Hum 22</td>
<td>Joe</td>
</tr>
<tr>
<td>CSE305</td>
<td>1</td>
<td>1234</td>
<td>Hum 22</td>
<td>Mary</td>
</tr>
</tbody>
</table>
Representation of Role in SQL

- Each role of relationship type produces a foreign key in corresponding relation
- Foreign key references table corresponding to entity type from which role values are drawn

Example 1

CREATE TABLE WorksIn (  
  Since DATE, -- attribute  
  Status CHAR (10), -- attribute  
  ProfId INTEGER, -- role (key of Professor)  
  DeptId CHAR (4), -- role (key of Department)  
  PRIMARY KEY (ProfId), -- since a professor works in at most one department  
  FOREIGN KEY (ProfId) REFERENCES Professor (Id),  
  FOREIGN KEY (DeptId) REFERENCES Department 
)
Example 2

CREATE TABLE Sold (  
Price INTEGER, -- attribute  
Date DATE, -- attribute  
ProjId INTEGER, -- role  
SupplierId INTEGER, -- role  
PartNumber INTEGER, -- role  
PRIMARY KEY (ProjId, SupplierId, PartNumber, Date),  
FOREIGN KEY (ProjId) REFERENCES Project,  
FOREIGN KEY (SupplierId) REFERENCES Supplier (Id),  
FOREIGN KEY (PartNumber) REFERENCES Part (Number) )

Representing Single-Role Key Constraints in Relational Model

- Key of the relation corresponding to the entity type is key of the relation corresponding to the relationship type
- Id is PK of Professor; ProfId is key of WorksIn
- Cannot use FK in Professor to refer to WorksIn since some professors (e.g., one with Id 4100) may not work in any dept
  - But ProfId is a foreign key in WorksIn that refers to Professor
Representing Type Hierarchies in the Relational Model

- Supertypes and subtypes can be realized as separate relations
  - Need a way of identifying subtype entity with its (unique) related supertype entity
    - Choose a CK and make it an attribute of all entity types in hierarchy

Type Hierarchies and the Relational Model

- Translated by adding the PK of supertype to all subtypes
- Make FK from subtypes to the supertype

```
<table>
<thead>
<tr>
<th>Id</th>
<th>attribs0</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>attribs1</th>
<th>Freshman</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>attribs2</th>
<th>Sophomore</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>attribs3</th>
<th>Junior</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>attribs4</th>
<th>Senior</th>
</tr>
</thead>
</table>
```

FOREIGN KEY `Id` REFERENCES Student in Freshman, Sophomore, Junior, Senior
Type Hierarchies and Relational Model

- Redundancy eliminated if **IsA** is not disjoint
  - For individuals who are both employees and students, Name and DOB are stored only once

<table>
<thead>
<tr>
<th>Person</th>
<th>Employee</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Name</td>
<td>DOB</td>
</tr>
<tr>
<td>1234</td>
<td>Mary</td>
<td>1950</td>
</tr>
<tr>
<td>SSN</td>
<td>Department</td>
<td>Salary</td>
</tr>
<tr>
<td>1234</td>
<td>Accounting</td>
<td>35000</td>
</tr>
<tr>
<td>SSN</td>
<td>GPA</td>
<td>StartDate</td>
</tr>
<tr>
<td>1234</td>
<td>3.5</td>
<td>1997</td>
</tr>
</tbody>
</table>

- If all disjoint,
  - Use a single relation (i.e., union of subtypes + a special column)
    - An example: UGRAD_YEAR with the domain of {'freshmen', 'sophomores', 'juniors', 'seniors'}

- If covering,
  - Create one relation per each subtype
  - Attributes of the relation
    - the union of attributes of the subtype and those of supertype
Discussed So Far...

- **Entity instance, Entity Type**
  - Attribute, domain, key, ...

- **Relationship instance, Relationship Type**
  - Relationship attribute, role, relationship key
  - Single-role key constraint (at most one, arrow)
  - Participation constraint (at least one, thick line)

- **Entity type hierarchy (IsA relationship)**
  - Covering, disjoint

- **Representation of E-R model into relational data model**
  - Multi-valued attribute
  - Single-role key constraint
  - IsA relationship