The Entity-Relationship Model

Part II

ER Cardinality Examples

ORDER (0,1) SALE (1,1) INVOICE

PERSON (1,1) RESIDENCE (0,N) CITY

TOURIST (1,N) RESERVATION (0,N) VOYAGE
Textbook Notation

...for (1,1) cardinalities (*key constraints*)

![Entity-Relationship Model Diagram]

Examples of Keys

*(internal) single-attribute key (unary key)*

- Slides Part 1 (UML inspired) notation
- Text notation
Examples of Keys

(Internal) multi-attribute (n-ary) key

**PERSON**
- DateOfBirth
- Surname
- FirstName
- Address

Examples of Keys

foreign, multi-attribute key (aka weak entity set)

**STUDENT**
- Registration
- Year
- Surname

**ENROLMENT**
- (1:1)

**UNIVERSITY**
- Name
- City
- Address

- Arrow on line indicates max cardinality of 1 (key)
- Double (or thick) line indicates min cardinality of 1
- aka participation constraint
An Entity Hierarchy

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

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?

- Note constraints of the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.
  - We’ll refine things in our logical (relational) design
Entity vs. Attribute

- Should *address* be an attribute of Employees or an entity (related to Employees)?
- **Depends** upon how we want to use 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, *address* must be modeled as an entity (since attribute values are atomic).

Entity vs. Attribute (Cont.)

- **Works_In2** 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.**

![Diagram of Entity-Relationship Model](The Entity-Relationship Model)
**Entity vs. Relationship**

OK as long as a manager gets a separate discretionary budget (dbudget) for each dept.

What if manager’s dbudget covers all managed depts?
(can repeat value, but such redundancy is problematic)

```
Courses database:
- Courses, Students, Professors
- Courses have ids, titles, credits. The id is unique.
- Courses have multiple sections that have time, a room and exactly one teacher
- Professors have a unique name
- Students take courses and receive a grade
- Students may repeat a course
- Must track students’ course schedules and transcripts including grades, semester taken, etc.
- Must track which classes a professor has taught
- Database should work over multiple semesters
```
Summary of Conceptual Design

- **Conceptual design** follows **requirements analysis**, yielding 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.
  - Note: There are many variations on ER model.
  - Both graphically and conceptually.

- **Basic constructs**: entities, relationships, and attributes (of entities and relationships).
- **Some additional constructs**: weak entities, ISA hierarchies, and aggregation.

Summary of ER (Cont.)

- Several kinds of integrity constraints:
  - *key constraints* (max cardinality 1)
  - *participation constraints* (min cardinality 1)
  - *overlap/covering* for ISA hierarchies.
- Some **foreign key constraints** are also implicit in the definition of a relationship set.
- Many other constraints (notably, *functional dependencies*) cannot be expressed.
- Constraints play an important role in determining the best database design for an enterprise.
Summary of ER (Cont.)

- **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, aggregation.
- **Ensuring good database design:** resulting relational schema should be analyzed and refined further.
  - Check for redundancy (see upcoming lectures)

ER to Relational Mapping
Logical DB Design: ER to Relational

- Entity sets to tables.

```
CREATE TABLE Employees
(ssn CHAR(11),
 name CHAR(20),
 lot INTEGER,
 PRIMARY KEY (ssn))
```

<table>
<thead>
<tr>
<th>ssn</th>
<th>name</th>
<th>lot</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
</tr>
</tbody>
</table>

Relationship Sets to Tables

- In translating a many-to-many 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.

```
CREATE TABLE Works_In
(ssn CHAR(1),
 did INTEGER,
 since DATE,
 PRIMARY KEY (ssn, did),
 FOREIGN KEY (ssn)
  REFERENCES Employees,
 FOREIGN KEY (did)
  REFERENCES Departments)
```

<table>
<thead>
<tr>
<th>ssn</th>
<th>did</th>
<th>since</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>51</td>
<td>1/1/91</td>
</tr>
<tr>
<td>123-22-3666</td>
<td>56</td>
<td>3/3/93</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>51</td>
<td>2/2/92</td>
</tr>
</tbody>
</table>
Review: Key Constraints

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

Translation to relational model?

Alternative notation: (0,N) left, (0, 1) right

Translating ER Diagrams with Key Constraints

- Map relationship set 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!)

```
(1,N)   (1,1)   (1,N)
Employees   Manages   Departments

(0,N)
Works_In

The Entity-Relationship Model -- 23
```

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!)

```
(0,N)   (1,1)   (1,N)
Employees   Manages   Departments

(1,N)
Works_In

The Entity-Relationship Model -- 24
```
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
)
```

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.

```
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

- Attributes are inherited
  - From superclass

- **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 Rel’nships

- If each policy is owned by just 1 employee:
  - **Key** constraint on Policies would mean policy can only cover 1 dependent!

Bad design

Better design
Binary vs. Ternary Relationships (Contd.)

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

ER Model Summary

- Usually easier to understand than Relational
- Expresses relationships clearly
- Rules to convert ER-diagrams to Relational Schema
- Some systems use ER-model for schema design
- Some people use ER-model as step before creating relational tables