Conceptual Database Design Using the Entity-Relationship (ER) Model

Overview of Database Design

- **Conceptual design**: (ER Model is used for this.)
  - What are the **entities** and **relationships** we need?

- **Logical design**:
  - Transform ER design to Relational Schema

- **Schema Refinement**: (Normalization)
  - Check relational schema for redundancies and related anomalies.

- **Physical Database Design and Tuning**:
  - Consider typical workloads; (sometimes) modify the database design; select file types and indexes.
Entity-Relationship Model is a different model from the Relational Model

- **Relation model** has:
  - tables (relations) with attributes, keys, foreign keys, domain definitions for attributes

- **Entity-Relationship model** has:
  - entities with attributes, keys, and domain definitions for attributes
  - relationships among entities with cardinality constraints (in the book they refer to key constraints and participation constraints)

Entity-Relationship Diagram (original syntax)
Entity-Relationship Diagram (original syntax)

Legend:
- **Entity**
- **Relationship**
- **Attribute Name**

Legend:
- **Entity**
- **Relationship**
- **Attribute Name**

Entities

- **Employee**
  - ssn
  - name
  - lot
- **Department**
  - code
  - name
- **Project**
  - number
  - name
  - start date
  - end date
  - budget
Relationships

Attributes
UML version of the same E-R Diagram
UML: Unified Modeling Language – for OO Design

Equivalent Relational Schema

Employee (ssn, name, lot, home-dept)
Project-team(ssn, number)
Department (id, name, manager)
Project (number, name, start-date, end-date, budget, sponsor)
Equivalent Relational Schema - with foreign keys shown

Employee (ssn, name, lot, home-dept)
Project-team(ssn, number)
Department (id, name, manager)
Project (number, name, start-date, end-date, budget, sponsor)

Notice that the many-to-many relationship must be represented in a (new) table.

Many-to-many relationships

ERD

Relational DB Diagram
Relationship Types

Each relationship allows related records to be connected

Allows a DB to REMEMBER a relationship

What data do we need to record a relationship?

We must indicate which employee and which department we want to have connected (for this relationship).

We need the key value for an employee and the key value for the department – stored together – to represent the relationship.
Cardinality Constraints on Relationships
How many entities can participate?

Employee
ssn
name
lot

Department
code
name

Manager

Project
number
name
start-date
end-date
budget

assigned

0..*
0..*
0..*
0..*
1..1
1..1
0..1
0..1

sponsor

CSE 514 Introduction to Database Systems, © Lois Delcambre 1995-2004
Some slides adapted from R. Ramakrishnan, with permission
An employee can have 0 or 1 home departments.

An department can have 0 to any number of employees.
Unified Modeling Language (UML):
Class Diagram

Employee
- ssn
- name
- lot

Departments
- code
- name

1..1 home 0..* lot

Employee
- ssn
- name
- lot

Departments
- code
- name

0..* home 0..1 lot

Employee
- ssn
- name
- lot

Departments
- code
- name

0..* home 1..* lot

Which one is right? We must discover the semantics of the application!

Various notation for “one-to-many”

<table>
<thead>
<tr>
<th>One</th>
<th>Many</th>
<th>Zero..One</th>
<th>One..Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>m</td>
<td>0..1</td>
<td>1..*</td>
</tr>
<tr>
<td>1*</td>
<td></td>
<td>0-1</td>
<td>1+</td>
</tr>
</tbody>
</table>

Maximum cardinalities only
Minimum and maximum cardinalities

CSE 514 Introduction to Database Systems, © Lois Delcambre 1995-2004
Various notations for “many-to-many”

- `many` to `many`: `1..*` to `1..*`
- `one..many` to `one..many`: `1..*` to `1..*`
- `maximum cardinalities only` to `minimum and maximum cardinalities`: `1+` to `1+`

Relationships can have attributes

- **E-R notation**
  - Employee
    - `ssn`
    - `name`
    - `lot`
  - Department
    - `code`
    - `name`
    - `start-date`
  - Home

- **UML notation**
  - Employee
    - `ssn`
    - `name`
    - `lot`
  - Department
    - `code`
    - `name`
    - `start-date`
  - Home
Try all three locations for the attributes: which one makes sense?

Relationships can have role names (in addition to the name of the relationship)
Example: reading role names

An employee manages 0 or 1 departments

Employee
- ssn
- name
- lot

1..1
manages

managed-by

Department
- code
- name

0..1
manages

role name

relationship name

Same entity can participate in different “roles” for the same relationship

E-R notation
- Employee
  - ssn
  - name
  - lot
  - supervisor
  - subordinate

UML notation
- Employee
  - ssn
  - name
  - lot
  - supervisor
  - subordinate
  - Reports-to

0..1
0..*
Weak Entities (and Identifying Relationships)

This is a weak entity.

It wouldn’t be in the database if the "strong" entity weren’t present.

Weak Entities and Identifying Relationships:
Alternative Notation

participation in the identifying relationship is required

partial key - must be concatenated with key for the strong entity (SSN)
Weak Entities and Identifying Relationships: Alternative Notation

<table>
<thead>
<tr>
<th>Department</th>
<th>*</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td></td>
<td>SSN</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>office</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>manager</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>home</td>
</tr>
<tr>
<td></td>
<td>1..1</td>
<td>Children</td>
</tr>
<tr>
<td></td>
<td>0..*</td>
<td></td>
</tr>
</tbody>
</table>

Cardinalities for an identifying relationship

Dependents

| Name | Date-of-birth |

Ternary vs. Binary Relationships

These two schemas are not equivalent!
When would we use a ternary relationship?
When would we use three binary relationships?
Binary vs. Ternary Relationships (Cont.)

- The ternary relationship means that a Supplier must be authorized to supply a particular part to a particular project. E.g., Office-Depot can supply laser printer paper to project 112. Office-Max can supply paper clips to Project 112. Office-Max can supply pencils to project 115. (But based on that much information, Office-Max can’t supply pencils to 112.)

- The three binary relationships each represent something distinct. A Supplier can be authorized to supply certain products (Office-Max can supply pencils). A Project can require certain products (112 needs pencils). And a Supplier can be authorized to supply a certain project. (Office-Max supplies 112)

Therefore: Office-Max can supply pencils to 112.

---

Duality: entity ↔ value and attribute ↔ relationship

<table>
<thead>
<tr>
<th>Project</th>
<th>Assignment</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-number</td>
<td>Manager</td>
<td>SSN</td>
</tr>
<tr>
<td>P-name</td>
<td></td>
<td>E-Name</td>
</tr>
<tr>
<td>Due-Date</td>
<td></td>
<td>Office</td>
</tr>
</tbody>
</table>

Should Office be an attribute of Employee? or a separate entity? Most attributes can be “promoted” to an entity and some entities can be “demoted” to an attribute value.

This explains why there are so many different ways to design a schema.
Entity vs. Value of an Attribute

What are some reasons to model Office as an entity?

• there’s a many-to-many relationship between employee and office

• there are other attributes of Office

• Office needs to participate in other relationships such as a relationship with furniture or telephones or network drops (located in the office)
Key Constraints - as described in the text
(limiting participation in relationship to at most 1 entity)
same as maximum multiplicity of 1 in UML

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

Participation Constraint - as in text:
when an entity MUST participate in a relationship

a Project has exactly one manager

a Project MUST have a manager
and there is at most 1 employee who is manager
Translating an ER Diagram to a Relational Schema

1. Translate each entity into a table, with keys.

- **Entity:**
  - can be represented as a table in the relational model
  - has a key … which becomes a key for the table

```
CREATE TABLE Employee
(SSN CHAR(11),
E-Name CHAR(20),
Office INTEGER,
PRIMARY KEY (SSN))
```
A DBMS may or may not allow multi-valued attributes. If it doesn’t,
2. Create a table for the multi-valued attribute.

How many offices can one employee have?

- **Just one**
  - Project (P-number, P-name, Due-Date)
  - Employee (SSN, E-Name, Office)

- **More than one**
  - Project (P-number, P-name, Due-Date)
  - Employee (SSN, E-Name)
  - Office-Assn (SSN, Office)

### Sample Data

<table>
<thead>
<tr>
<th>Project</th>
<th>Employee</th>
<th>Office-Assn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Just one**
  - 12 Smith O-105
  - 15 Wei O-110
  - 20 Jones O-112

- **More than one**
  - 12 Smith
  - 15 Wei

  Office-Assn (SSN, Office)
  - 12 O-105
  - 12 O-106
  - 15 O-110
3. Translate each many to many relationship into a table

What are the attributes and what is the key for Assignment?

Assignment (?)

Answer: Assignment (P-Number, SSN)

P-Number is a foreign key for Project
SSN is a foreign key for Employee
What should we do with each one to many relationship?

Manager (?)

Project (P-number, P-name, Due-Date)
Employee (SSN, E-Name, Office)

4. Create a foreign key for a 1-to-many relationship.

Manager is a foreign key (referencing the Employee relation)

value of Manager must match an SSN
vs.

4. Or...Create a table for a 1-many relationship.

Project (P-number, P-name, Due-Date)
Employee (SSN, E-Name, Office)
Manager (P-number, SSN)

What are the tradeoffs between these two?

Note: P-number is the key for Manager

vs.

Project (P-number, P-name, Due-Date)
Employee (SSN, E-Name, Office, Managed-project)
Manager (P-number, SSN)

What if SSN is the key for Manager?
What do we do when a many-to-many relationship has an attribute?

Assignment (A-project, A-SSN, role, start-date, end-date)
Project (P-number, P-name, Due-Date)
Employee (SSN, E-Name, Office)
What do we do when a 1-to-many relationship has an attribute?

**Project** (P-number, P-name, Due-Date, Manager, start-date, end-date)

**Employee** (SSN, E-Name, Office)

Is this a good idea?
Participation Constraints in SQL

- We can require an one entity to be in a binary relationship using a foreign key which is required to be NOT NULL (but little else without resorting to CHECK constraints)

```sql
CREATE TABLE Department (
    d-code INTEGER,
    d-name CHAR(20),
    manager-ssn CHAR(9) NOT NULL,
    since DATE,
    PRIMARY KEY (d-code),
    FOREIGN KEY (manager-ssn) REFERENCES Employee,
    ON DELETE NO ACTION)
```

Translating Weak Entity Sets

- Weak entity and identifying relationship are translated into a single table. Must include key of strong entity, as a foreign key.
- When the owner entity is deleted, all owned weak entities must also be deleted.

```sql
CREATE TABLE Insurance_Policy (
    dep-name CHAR(20),
    age INTEGER,
    cost REAL,
    ssn CHAR(11) NOT NULL,
    PRIMARY KEY (dep-name, ssn),
    FOREIGN KEY (ssn) REFERENCES Employee,
    ON DELETE CASCADE)
```
Note ERDs can be at two levels:

the ERD level

and

the Relational Table level.

The difference is primarily with the many-to-many relationships.
Equivalent Relational Schema

Notice that the relationships shown in this diagram aren’t really needed. foreign keys reference other tables.

Summary of Translation Steps: ER to Tables

1. Create table for each entity; include single-valued attributes. Choose key.
2. Create table for each weak entity type; include single-valued attributes. Include key of owner as a foreign key in the weak entity. Set key as foreign key of owner plus local, partial key.
3. For each 1:1 relationship, add a foreign key to one of the entities involved in the relationship (a foreign key to the other entity in the relationship).*
4. For each 1:N relationship, add a foreign key to the entity on the N-side of the relationship (to reference the entity on the 1-side of the relationship).*
5. For each M:N relationship, create a new table. Include a foreign key for each participant entity in the relationship. The key for the new table is the set of all such foreign keys.
6. For each multi-valued attribute, construct a separate table. Repeat the key for the entity in this new table. It will serve as both the key for this table as well as a foreign key to the original table for the entity.

* Unless relationship has attributes. If it does, create new table for relationship.

This algorithm from Elmasri/Navathe, p. 174.