Chapter 4
Logical Database Design and the Relational Model

Objectives

• Define terms for the relational data model
• Transform EE-R diagrams to relations
• Create tables with entity and relational integrity constraints
Steps in Database Problem Solving

Logical Model: Relational Model

- Can represent all kinds of information
- Based on Math (relations)
- Natural to people
- Relatively simple
- We know how to implement it fast
Components of Relational Model

• Data structure
  – Tables (relations), rows, columns

• Data manipulation
  – Powerful SQL operations for retrieving and modifying data

• Data integrity
  – Mechanisms for implementing business rules that maintain integrity of manipulated data

Motivating Example

• Make a list of students in the class, keeping their ID, name and phone number
Motivating Example

• Make a list of students in the class, keeping their ID, name and phone number
• You’d probably come up with something like this:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>Mike</td>
<td>111</td>
</tr>
<tr>
<td>yy</td>
<td>Elisa</td>
<td>222</td>
</tr>
</tbody>
</table>

• This is the basic structure of the relational model, a table or relation

Extra Assumptions

• You would not repeat the same row twice
• No two rows have the same ID, but they may have the same name and phone number

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>Mike</td>
<td>111</td>
</tr>
<tr>
<td>yy</td>
<td>Elisa</td>
<td>222</td>
</tr>
</tbody>
</table>

• ID would be the PRIMARY KEY (PK).
Now add emails … (many!)

- Now you need to add the emails of each student, but you do not know how many emails
- Can you come up with a solution? Try it …

Many Fields

- Could come up with something like this

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Phone</th>
<th>Email1</th>
<th>Email2</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>Mike</td>
<td>111</td>
<td>bad</td>
<td>idea 😞</td>
</tr>
<tr>
<td>yy</td>
<td>Elisa</td>
<td>222</td>
<td>bad</td>
<td>idea 😊</td>
</tr>
</tbody>
</table>

- Above would not work very well. How many fields?
  - Wasted space
  - What if a student has more emails?
  - How to access the emails?
Un-Normalized

- Could also try this:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Phone</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>Mike</td>
<td>111</td>
<td><a href="mailto:mk@ad.com">mk@ad.com</a></td>
</tr>
<tr>
<td>xx</td>
<td>Mike</td>
<td>111</td>
<td><a href="mailto:mk@vu.edu">mk@vu.edu</a></td>
</tr>
<tr>
<td>yy</td>
<td>Elisa</td>
<td>222</td>
<td><a href="mailto:eli@vu.edu">eli@vu.edu</a></td>
</tr>
</tbody>
</table>

- Problem is duplication, we are repeating the name and phone number in the second row
  - What if Mike changes his phone?

- Later we will study normalization to solve this.

Now add emails … (many!)

- A much better way:

<table>
<thead>
<tr>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>StudentID</td>
</tr>
<tr>
<td>xx</td>
</tr>
<tr>
<td>xx</td>
</tr>
<tr>
<td>yy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>xx</td>
</tr>
<tr>
<td>yy</td>
</tr>
</tbody>
</table>

- Every StudentID on the second table needs a matching ID on the first table: StudentID is a FOREIGN KEY
- In a way, StudentID in the second table is a pointer or reference to the first table
Formalizing: Relations

- Definition: A relation is a **named table** of data
  - Table is made up of rows (records or tuples), and columns (attributes or fields)
- Requirements for a table to be a relation:
  1. Has a unique name.
  2. Every attribute value is atomic (not multivalued or composite)
  3. Every row is unique
  4. Attributes (columns) in tables have unique names
  5. The order of the columns is irrelevant
  6. The order of the rows is irrelevant

By definition, all relations are in **1st Normal Form (1NF)**.

Correspondence with ER Model

- Relations (tables) correspond to entity types and to many-to-many relationship types
- Rows correspond to entity instances and to many-to-many relationship instances
- Columns correspond to attributes

- NOTE: The word **relation** (in relational database) is NOT the same as the word **relationship** (in ER model)
Formalizing Key Fields

- **Primary key (PK)**
  - Minimal set of attributes that uniquely identifies a row, chosen for referencing
  - This is how we can guarantee that all rows are unique

- **Foreign key (FK)**
  - Set of attributes in a table that serves as a reference to the primary key of another table

- Keys can be simple or composite
- Used as indexes to speed up queries

Figure 4-3 Schema for four relations (Pine Valley Furniture Company)

* Not in Figure 2-22 for simplicity.
Key Constraints

• **Entity Integrity Constraint**
  – No attribute of the PK may be null

• **Referential Integrity Constraint**
  – For a FK, either all attributes are null, or the values appear in the PK of a row of the referred table

![Referential integrity constraints (Pine Valley Furniture)](image)
Referential integrity constraints are implemented with foreign key to primary key references.

Key Constraints – Example

• Delete Rules
  – **Restrict** – don’t allow delete of “parent” side if related rows exist in “dependent” side
  – **Cascade** – automatically delete “dependent” side rows that correspond with the “parent” side row to be deleted
  – **Set-to-Null** – set the foreign key in the dependent side to null if deleting from the parent side → not allowed for weak entities
Transforming E-R Into Relations

- Use a rectangle for each entity (table), with attributes inside rectangles, too
  - Can be vertical or horizontal
  - Primary key is underlined
- Use arrows from Foreign key to Primary key

![Diagram of E-R to Relations transformation]

```
Student
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Student
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>StudentID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Email
<table>
<thead>
<tr>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Phone</td>
</tr>
</tbody>
</table>
```
E-R vs. Relational

• Entities are represented by tables
  – But tables may also represent relationships, or multivalued attributes

• Foreign Keys used to relate table rows
  – Similar to relationships in E-R, but lower level

• Relational model is more concrete, lower level
  – Usually many more tables than entities
  – Harder to understand by non-technical people
  – Directly implementable

Six Cases of Transforming E-R Diagrams into Relations

1. Map Regular Entities
2. Map Binary Relationships
3. Map Weak Entities
4. Map Associative Entities
5. Map Unary Relationships
6. Map Ternary (and n-ary) Relationships
1. Mapping Regular Entities

- Create a new table for each entity
- Remember to underline the identifier
- For composite attributes, map only the basic pieces
- Derived attributes disappear
- For multivalued attributes we need a new table
- We may need to create several tables for independent multivalued attributes

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>SSN</th>
<th>Name (First, Middle, Last)</th>
<th>{Emails}</th>
<th>Date of Birth</th>
<th>[Age]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
</tr>
<tr>
<td>First</td>
</tr>
<tr>
<td>Middle</td>
</tr>
<tr>
<td>Last</td>
</tr>
<tr>
<td>DoB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Email</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>BOOK</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISBN</td>
</tr>
<tr>
<td>Title</td>
</tr>
<tr>
<td>{Authors}</td>
</tr>
<tr>
<td>Format (Binding, NumPages, Dimensions, [Weight])</td>
</tr>
</tbody>
</table>
Six Cases of Transforming E-R Diagrams into Relations

1. Map Regular Entities
2. Map Binary Relationships
3. Map Weak Entities
4. Map Associative Entities
5. Map Unary Relationships
6. Map Ternary (and n-ary) Relationships

Mapping Binary Relationships

- **One-to-Many**
  - Primary key on the one side becomes a foreign key on the many side (Fig. 4-12).

- **One-to-One**
  - Primary key on the mandatory side becomes a foreign key on the optional side (Fig. 4-14).

- **Many-to-Many**
  - Create a new relation with the primary keys of the two entities as its primary key (Fig. 4-13).
Fig. 4-12: Example of mapping a 1:M relationship
(a) Relationship between customers and orders

Note the mandatory one

Fig. 4-12: (b) Mapping the relationship

[Primary key on the one side becomes a foreign key on the many side]

Again, no null value in the foreign key...this is because of the mandatory minimum cardinality

You Try – 1:M Relationship
Many-to-Many Relationship

- For a many-to-many, we need a new table representing the relationship.
- This table has Foreign Keys to both entities.
Figure 4-13 Example of mapping an M:N relationship

a) Completes relationship (M:N)

The Completes relationship will need to become a separate relation.

Figure 4-13 Example of mapping an M:N relationship (cont.)

b) Three resulting relations

The EMPLOYEE relation is now a Composite primary key (CPK) relation.

The EMPLOYEE relation contains fields EmployeeID, EmployeeName, and EmployeeBirthDate.

The COURSE relation contains fields CourseID and CourseTitle.

The CERTIFICATE relation contains fields EmployeeID, CourseID, and DateCompleted.
Six Cases of Transforming E-R Diagrams into Relations

1. Map Regular Entities
2. Map Binary Relationships
3. Map Weak Entities
4. Map Associative Entities
5. Map Unary Relationships
6. Map Ternary (and n-ary) Relationships
3. Mapping Weak Entities

- A weak entity becomes a separate relation with a foreign key taken from the strong entity.
- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)

Weak Entities

- Transform the strong entity normally.
- For the weak entity, the PK becomes the identifier, plus the PK of the identifying entity.
Six Cases of Transforming E-R Diagrams into Relations

1. Map Regular Entities
2. Map Binary Relationships
3. Map Weak Entities
4. Map Associative Entities
5. Map Unary Relationships
6. Map Ternary (and n-ary) Relationships
4. Mapping Associative Entities

- Identifier Not Assigned
  - Default primary key for the association relation is composed of the primary keys of the two entities (as in M:N relationship)

- Identifier Assigned
  - It is natural and familiar to end-users
  - Default identifier may not be unique

Figure 4-15: Mapping an associative entity
(a) Associative entity (ORDER LINE) [Default primary key for the association relation is NOT assigned]
Figure 4-16: Mapping an associative entity

(a) Associative entity (SHIPMENT)

(b) Three resulting relations

You Try …
Six Cases of Transforming E-R Diagrams into Relations

1. Map Regular Entities
2. Map Binary Relationships
3. Map Weak Entities
4. Map Associative Entities
5. Map Unary Relationships
6. Map Ternary (and n-ary) Relationships

5. Mapping Unary Relationships

- Same as other relationships, except that the FK may go to the same table.
- For one-to-many, the table has a reference to other rows of the same table.
- For many-to-many, an extra table has two FKs, both to the same table (Fig. 4-18).
Figure 4-17 Mapping a unary 1:N relationship

(a) EMPLOYEE entity with unary relationship

(b) EMPLOYEE relation with recursive foreign key

A recursive FK is a FK in a relation that references the PK values of that same relation.

Figure 4-18: Mapping a unary M:N relationship

(a) Bill-of-materials relationships (M:N)

(b) ITEM and COMPONENT relations

One table for the entity type.

One table for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity.
You Try – 1:M Unary

You Try – M:N Unary
Six Cases of Transforming E-R Diagrams into Relations

1. Map Regular Entities
2. Map Binary Relationships
3. Map Weak Entities
4. Map Associative Entities
5. Map Unary Relationships
6. Map Ternary (and n-ary) Relationships

6. Mapping Ternary Relationships

- One relation for each entity and one for the associative entity.
- Associative entity has foreign keys to each entity in the relationship
Figure 4-19 Mapping a ternary relationship

a) PATIENT TREATMENT Ternary relationship with associative entity

Remember that the primary key MUST be unique.

This is why treatment date and time are included in the composite primary key.

But this makes a very cumbersome key...

It would be better to create a surrogate key like Treatment#.

(A patient may receive a treatment once in the morning, then the same treatment in the afternoon.)
From EE-R Diagrams to Relations (Tables)

EE-R to Relations

- Mapping Supertype/Subtype Relationships
- One relation for supertype and for each subtype
- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established between supertype and each subtype, with supertype as primary table
Figure 4-20 Supertype/subtype relationship

These are implemented as one-to-one relationships.

Figure 4-21 Mapping supertype/subtype relationships to relations
Relational Model

Practice Exercises

#1

<table>
<thead>
<tr>
<th>TV SERIES</th>
<th>EPISODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Number</td>
</tr>
<tr>
<td>Title</td>
<td>Title</td>
</tr>
<tr>
<td></td>
<td>FirstAiring Date</td>
</tr>
</tbody>
</table>
#2

COURSE
ID
Name

SECTION
SecNo
Term(Semester, Year)

TEACHER
SSN
Name

Belongs

Teaches

#3

SONG
Song_ID
Song_Title
Song_Length

CD
CD_ID
CD_Title

Includes

Perform

PERSON
Person_ID
Person_Name(First, Last)

Authors

Performs
Next Topic

- Next topic is considered the most important theory in database management.
- What is it?
- **Normalization**

Table 4-6: Preview of Normalization

Below is a list of parking tickets issued by the Public Safety office to vehicles parked illegally on campus.

**How would you organize such data into relations?**

<table>
<thead>
<tr>
<th>St ID</th>
<th>L Name</th>
<th>F Name</th>
<th>Phone No</th>
<th>St Lic</th>
<th>Lic No</th>
<th>Ticket #</th>
<th>Date</th>
<th>Code</th>
<th>Fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>38249</td>
<td>Brown</td>
<td>Thomas</td>
<td>111-7804</td>
<td>FL</td>
<td>BRY 123</td>
<td>15634</td>
<td>10/17/10</td>
<td>2</td>
<td>$25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16017</td>
<td>11/13/10</td>
<td>1</td>
<td>$15</td>
</tr>
<tr>
<td>82453</td>
<td>Green</td>
<td>Sally</td>
<td>391-1689</td>
<td>AL</td>
<td>TRE 141</td>
<td>14987</td>
<td>10/05/10</td>
<td>3</td>
<td>$100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16293</td>
<td>11/18/10</td>
<td>1</td>
<td>$15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>17892</td>
<td>12/13/10</td>
<td>2</td>
<td>$25</td>
</tr>
</tbody>
</table>