Chapter 5: Logical Database Design and the Relational Model

*Modern Database Management*

*9th Edition*

Jeffrey A. Hoffer, Mary B. Prescott, Heikki Topi

Objectives

- Definition of terms
- List five properties of relations
- State two properties of candidate keys
- Define first, second, and third normal form
- Describe problems from merging relations
- Transform E-R and EER diagrams to relations
- Create tables with entity and relational integrity constraints
- Use normalization to convert anomalous tables to well-structured relations
Relation

- **Definition**: A relation is a named, two-dimensional table of data.
- **Table consists of rows (records) and columns (attribute or field)**.
- **Requirements for a table to qualify as a relation**:
  - It must have a unique name.
  - Every attribute value must be atomic (not multivalued, not composite).
  - Every row must be unique (can’t have two rows with exactly the same values for all their fields).
  - Attributes (columns) in tables must have unique names.
  - The order of the columns must be irrelevant.
  - The order of the rows must be irrelevant.

**NOTE**: all relations are in **1st Normal form**.
Correspondence with E-R Model

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

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

- Keys are special fields that serve two main purposes:
  - **Primary keys** are unique identifiers of the relation in question. Examples include employee numbers, social security numbers, etc. *This is how we can guarantee that all rows are unique.*
  - **Foreign keys** are identifiers that enable a dependent relation (on the many side of a relationship) to refer to its parent relation (on the one side of the relationship).
- Keys can be **simple** (a single field) or **composite** (more than one field).
- Keys usually are used as indexes to speed up the response to user queries. (More on this in Chapter 6)
Figure 5-3 Schema for four relations (Pine Valley Furniture Company)

<table>
<thead>
<tr>
<th>CUSTOMER</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer_ID</td>
<td>Customer_Name</td>
<td>Customer_Address</td>
<td>City</td>
<td>State</td>
</tr>
</tbody>
</table>

**Primary Key**

<table>
<thead>
<tr>
<th>ORDER</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Order_ID</td>
<td>Order_Date</td>
<td>Customer_ID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Foreign Key**

(implements 1:N relationship between customer and order)

<table>
<thead>
<tr>
<th>ORDER LINE</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Order_ID</td>
<td>Product_ID</td>
<td>Ordered_Quantity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Combined, these are a *composite primary key* (uniquely identifies the order line)...individually they are *foreign keys* (implement M:N relationship between order and product)

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Product_ID</td>
<td>Product_Description</td>
<td>Product_Finish</td>
<td>Standard_Price</td>
<td>Product_Line_ID</td>
</tr>
</tbody>
</table>

* Not in Figure 5-22 for simplicity.
Integrity Constraints

- **Domain Constraints**
  - Allowable values for an attribute. See Table 5-1

- **Entity Integrity**
  - No primary key attribute may be null. All primary key fields **MUST** have data

- **Action Assertions**
  - Business rules. Recall from Chapter 4
Domain definitions enforce domain integrity constraints

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>DOMAIN NAME</th>
<th>DESCRIPTION</th>
<th>DOMAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer_ID</td>
<td>Customer_IDS</td>
<td>Set of all possible customer IDs</td>
<td>character: size 5</td>
</tr>
<tr>
<td>Customer_Name</td>
<td>Customer_Names</td>
<td>Set of all possible customer names</td>
<td>character: size 25</td>
</tr>
<tr>
<td>Customer_Address</td>
<td>Customer_Addresses</td>
<td>Set of all possible customer addresses</td>
<td>character: size 30</td>
</tr>
<tr>
<td>City</td>
<td>Cities</td>
<td>Set of all possible cities</td>
<td>character: size 20</td>
</tr>
<tr>
<td>State</td>
<td>States</td>
<td>Set of all possible states</td>
<td>character: size 2</td>
</tr>
<tr>
<td>Postal_Code</td>
<td>Postal_Codes</td>
<td>Set of all possible postal zip codes</td>
<td>character: size 10</td>
</tr>
<tr>
<td>Order_ID</td>
<td>Order_IDs</td>
<td>Set of all possible order IDs</td>
<td>character: size 5</td>
</tr>
<tr>
<td>Order_Date</td>
<td>Order_Dates</td>
<td>Set of all possible order dates</td>
<td>date format mm/dd/yy</td>
</tr>
<tr>
<td>Product_ID</td>
<td>Product_IDs</td>
<td>Set of all possible product IDs</td>
<td>character: size 5</td>
</tr>
<tr>
<td>Product_Description</td>
<td>Product_Descriptions</td>
<td>Set of all possible product descriptions</td>
<td>character size 25</td>
</tr>
<tr>
<td>Product_Finish</td>
<td>Product_Finishes</td>
<td>Set of all possible product finishes</td>
<td>character: size 15</td>
</tr>
<tr>
<td>Standard_Price</td>
<td>Unit_Prices</td>
<td>Set of all possible unit prices</td>
<td>monetary: 6 digits</td>
</tr>
<tr>
<td>Product_Line_ID</td>
<td>Product_Line_IDs</td>
<td>Set of all possible product line IDs</td>
<td>integer: 3 digits</td>
</tr>
<tr>
<td>OrderedQuantity</td>
<td>Quantities</td>
<td>Set of all possible ordered quantities</td>
<td>integer: 3 digits</td>
</tr>
</tbody>
</table>

Chapter 5 © 2009 Pearson Education, Inc. Publishing as Prentice Hall
Integrity Constraints

- Referential Integrity - rule states that any foreign key value (on the relation of the many side) MUST match a primary key value in the relation of the one side. (Or the foreign key can be null)

- For 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
Referential integrity constraints (Pine Valley Furniture)

Referential integrity constraints are drawn via arrows from dependent to parent table.

<table>
<thead>
<tr>
<th>CUSTOMER</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer_ID</td>
<td>Customer_Name</td>
<td>Customer_Address</td>
<td>City</td>
<td>State</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORDER</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Order_ID</td>
<td>Order_Date</td>
<td>Customer_ID</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORDER LINE</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Order_ID</td>
<td>Product_ID</td>
<td>Ordered_Quantity</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Product_ID</td>
<td>Product_Description</td>
<td>Product_Finish</td>
<td>Standard_Price</td>
<td>Product_Line_ID</td>
<td></td>
</tr>
</tbody>
</table>
Referential integrity constraints are implemented with foreign key to primary key references.
Transforming EER Diagrams into Relations

Mapping Regular Entities to Relations

1. Simple attributes: E-R attributes map directly onto the relation
2. Composite attributes: Use only their simple, component attributes
3. Multivalued Attribute: Becomes a separate relation with a foreign key taken from the superior entity
Figure 5-8 Mapping a regular entity

(a) CUSTOMER entity type with simple attributes

(b) CUSTOMER relation
Figure 5-9 Mapping a composite attribute

(a) CUSTOMER entity type with composite attribute

(b) CUSTOMER relation with address detail
Figure 5-10 Mapping an entity with a multivalued attribute

(a) EMPLOYEE
Employee_ID
Employee_Name
Employee_Address
{Skill}

(b) Multivalued attribute becomes a separate relation with foreign key

EMPLOYEE
Employee_ID Employee_Name Employee_Address

EMPLOYEE_SKILL
Employee_ID Skill

One-to-many relationship between original entity and new relation
Transforming EER Diagrams into Relations (cont.)

Mapping Weak Entities

- Becomes a separate relation with a foreign key taken from the superior entity
- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)
Figure 5-11 Example of mapping a weak entity

a) Weak entity DEPENDENT
b) Relations resulting from weak entity

NOTE: the domain constraint for the foreign key should NOT allow null value if DEPENDENT is a weak entity

Composite primary key
Transforming EER Diagrams into Relations (cont.)

Mapping Binary Relationships

- One-to-Many–Primary key on the one side becomes a foreign key on the many side
- Many-to-Many–Create a \textit{new relation} with the primary keys of the two entities as its primary key
- One-to-One–Primary key on the mandatory side becomes a foreign key on the optional side
Figure 5-12 Example of mapping a 1:M relationship

a) Relationship between customers and orders

Note the mandatory one

b) Mapping the relationship

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

Chapter 5 © 2009 Pearson Education, Inc. Publishing as Prentice Hall
Figure 5-13 Example of mapping an M:N relationship

a) Completes relationship (M:N)

The *Completes* relationship will need to become a separate relation.
Figure 5-13 Example of mapping an M:N relationship (cont.)

b) Three resulting relations

- **EMPLOYEE**
  - Employee_ID
  - Employee_Name
  - Birth_Date

- **CERTIFICATE**
  - Employee_ID
  - Course_ID
  - Date_Completed

- **COURSE**
  - Course_ID
  - Course_Title

Composite primary key

New intersection relation

Foreign key

New foreign key
Figure 5-14 Example of mapping a binary 1:1 relationship

a) In_charge relationship (1:1)

Often in 1:1 relationships, one direction is optional
b) Resulting relations

Foreign key goes in the relation on the optional side, matching the primary key on the mandatory side.
Transforming EER Diagrams into Relations (cont.)

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 5-15 Example of mapping an associative entity

a) An associative entity
b) Three resulting relations

ORDER

<table>
<thead>
<tr>
<th>Order_ID</th>
<th>Order_Date</th>
</tr>
</thead>
</table>

ORDER LINE

<table>
<thead>
<tr>
<th>Order_ID</th>
<th>Product_ID</th>
<th>Ordered_Quantity</th>
</tr>
</thead>
</table>

PRODUCT

<table>
<thead>
<tr>
<th>Product_ID</th>
<th>Product_Description</th>
<th>Product_Finish</th>
<th>Standard_Price</th>
<th>Product_Line_ID</th>
</tr>
</thead>
</table>

Composite primary key formed from the two foreign keys
Figure 5-16 Example of mapping an associative entity with an identifier

a) SHIPMENT associative entity
Figure 5-16 Example of mapping an associative entity with an identifier (cont.)

b) Three resulting relations

- **CUSTOMER**
  - Customer_ID
  - Customer_Name

- **SHIPMENT**
  - Shipment_ID
  - Customer_ID
  - Vendor_ID
  - Shipment_Date

- **VENDOR**
  - Vendor_ID
  - Vendor_Address

Primary key differs from foreign keys
Transforming EER Diagrams into Relations (cont.)

Mapping Unary Relationships

- One-to-Many–Recursive foreign key in the same relation
- Many-to-Many–Two relations:
  - One for the entity type
  - One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity
Figure 5-17 Mapping a unary 1:N relationship

(a) EMPLOYEE entity with unary relationship

(b) EMPLOYEE relation with recursive foreign key

Chapter 5 © 2009 Pearson Education, Inc. Publishing as Prentice Hall
Figure 5-18 Mapping a unary M:N relationship

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

(b) ITEM and COMPONENT relations
Transforming EER Diagrams into Relations (cont.)

Mapping Ternary (and n-ary) Relationships

- One relation for each entity and one for the associative entity
- Associative entity has foreign keys to each entity in the relationship
Figure 5-19 Mapping a ternary relationship

a) PATIENT TREATMENT Ternary relationship with associative entity
b) Mapping the ternary relationship PATIENT TREATMENT

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#.

Chapter 5
Transforming EER Diagrams into Relations (cont.)

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 5-21
Mapping Supertype/subtype relationships to relations

These are implemented as one-to-one relationships
Data Normalization

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that *avoid unnecessary duplication of data*.

- The process of decomposing relations with anomalies to produce smaller, *well-structured* relations.
Well-Structured Relations

- A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies.

- Goal is to avoid anomalies:
  - **Insertion Anomaly**—adding new rows forces user to create duplicate data.
  - **Deletion Anomaly**—deleting rows may cause a loss of data that would be needed for other future rows.
  - **Modification Anomaly**—changing data in a row forces changes to other rows because of duplication.

General rule of-thumb: A table should not pertain to more than one entity type.
Question–Is this a relation? Answer–Yes: Unique rows and no multivalued attributes

Question–What’s the primary key? Answer–Composite: Emp_ID, Course_Title
Anomalies in this Table

- **Insertion**—can’t enter a new employee without having the employee take a class
- **Deletion**—if we remove employee 140, we lose information about the existence of a Tax Acc class
- **Modification**—giving a salary increase to employee 100 forces us to update multiple records

Why do these anomalies exist?
Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities.
Functional Dependencies and Keys

- **Functional Dependency**: The value of one attribute (the *determinant*) determines the value of another attribute.

- **Candidate Key**: A unique identifier. One of the candidate keys will become the primary key.
  - E.g. perhaps there is both credit card number and SS# in a table...in this case both are candidate keys.
  - Each non-key field is functionally dependent on every candidate key.
Figure 5.22 Steps in normalization

1. Table with multivalued attributes
   - First normal form
2. Remove multivalued attributes
3. Second normal form
   - Remove partial dependencies
4. Third normal form
   - Remove transitive dependencies
5. Remove remaining anomalies resulting from multiple candidate keys
   - Boyce-Codd normal form
6. Fourth normal form
   - Remove multivalued dependencies
7. Fifth normal form
   - Remove remaining anomalies

Chapter 5 © 2009 Pearson Education, Inc. Publishing as Prentice Hall
First Normal Form

- No multivalued attributes
- Every attribute value is atomic
- Fig. 5-25 *is not* in 1st Normal Form (multivalued attributes) → it is not a relation
- Fig. 5-26 *is* in 1st Normal form
- *All relations* are in 1st Normal Form
Table with multivalued attributes, not in 1\textsuperscript{st} normal form

Note: this is NOT a relation
Table with no multivalued attributes and unique rows, in 1st normal form

Table with no multivalued attributes and unique rows, in 1st normal form

<table>
<thead>
<tr>
<th>Order_ID</th>
<th>Order_Date</th>
<th>Customer_ID</th>
<th>Customer_Name</th>
<th>Customer_Address</th>
<th>Product_ID</th>
<th>Product_Description</th>
<th>Product_Finish</th>
<th>Unit_Price</th>
<th>Ordered_Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1006</td>
<td>10/24/2008</td>
<td>2</td>
<td>Value Furniture</td>
<td>Plano, TX</td>
<td>7</td>
<td>Dining Table</td>
<td>Natural Ash</td>
<td>800.00</td>
<td>2</td>
</tr>
<tr>
<td>1006</td>
<td>10/24/2008</td>
<td>2</td>
<td>Value Furniture</td>
<td>Plano, TX</td>
<td>5</td>
<td>Writer's Desk</td>
<td>Cherry</td>
<td>325.00</td>
<td>2</td>
</tr>
<tr>
<td>1006</td>
<td>10/24/2008</td>
<td>2</td>
<td>Value Furniture</td>
<td>Plano, TX</td>
<td>4</td>
<td>Entertainment Center</td>
<td>Natural Maple</td>
<td>650.00</td>
<td>1</td>
</tr>
<tr>
<td>1007</td>
<td>10/25/2008</td>
<td>6</td>
<td>Furniture Gallery</td>
<td>Boulder, CO</td>
<td>11</td>
<td>4-Dr Dresser</td>
<td>Oak</td>
<td>500.00</td>
<td>4</td>
</tr>
<tr>
<td>1007</td>
<td>10/25/2008</td>
<td>6</td>
<td>Furniture Gallery</td>
<td>Boulder, CO</td>
<td>4</td>
<td>Entertainment Center</td>
<td>Natural Maple</td>
<td>650.00</td>
<td>3</td>
</tr>
</tbody>
</table>

Note: this is a relation, but not a well-structured one
Anomalies in this Table

- **Insertion**—if new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication

- **Deletion**—if we delete the Dining Table from Order 1006, we lose information concerning this item's finish and price

- **Update**—changing the price of product ID 4 requires update in several records

Why do these anomalies exist?

Because there are multiple themes (entity types) in one relation. This results in duplication and an unnecessary dependency between the entities
Second Normal Form

- 1NF PLUS *every non-key attribute is fully functionally dependent on the ENTIRE primary key*
  - Every non-key attribute must be defined by the entire key, not by only part of the key
  - No partial functional dependencies
Figure 5-27 Functional dependency diagram for INVOICE

Order_ID \(\rightarrow\) Order_Date, Customer_ID, Customer_Name, Customer_Address
Customer_ID \(\rightarrow\) Customer_Name, Customer_Address
Product_ID \(\rightarrow\) Product_Description, Product_Finish, Unit_Price
Order_ID, Product_ID \(\rightarrow\) Order_Quantity

Therefore, NOT in 2\textsuperscript{nd} Normal Form
Partial dependencies are removed, but there are still transitive dependencies.
Third Normal Form

- 2NF PLUS \textit{no transitive dependencies} (functional dependencies on non-primary-key attributes)
- Note: This is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third
- Solution: Non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as foreign key in the old table
Figure 5-29 Removing partial dependencies

Transitive dependencies are removed

ORDER (3NF)

CUSTOMER (3NF)
Merging Relations

- View Integration—Combining entities from multiple ER models into common relations

- Issues to watch out for when merging entities from different ER models:
  - Synonyms—two or more attributes with different names but same meaning
  - Homonyms—attributes with same name but different meanings
  - Transitive dependencies—even if relations are in 3NF prior to merging, they may not be after merging
  - Supertype/subtype relationships—may be hidden prior to merging
Enterprise Keys

- Primary keys that are unique in the whole database, not just within a single relation
- Corresponds with the concept of an object ID in object-oriented systems
Figure 5-31 Enterprise keys

a) Relations with enterprise key

b) Sample data with enterprise key

---

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>OID</th>
<th>Object_Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>EMPLOYEE</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>CUSTOMER</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>CUSTOMER</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>EMPLOYEE</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>EMPLOYEE</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>CUSTOMER</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>CUSTOMER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>OID</th>
<th>Emp_ID</th>
<th>Emp_Name</th>
<th>Dept_Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>100</td>
<td>Jennings, Fred</td>
<td>Marketing</td>
<td>50000</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>101</td>
<td>Hopkins, Dan</td>
<td>Purchasing</td>
<td>45000</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>102</td>
<td>Huber, Ike</td>
<td>Accounting</td>
<td>45000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CUSTOMER</th>
<th>OID</th>
<th>Cust_ID</th>
<th>Cust_Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>100</td>
<td>Fred’s Warehouse</td>
<td>Greensboro, NC</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>101</td>
<td>Bargain Bonanza</td>
<td>Moscow, ID</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>102</td>
<td>Jasper’s</td>
<td>Tallahassee, FL</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>100</td>
<td>Desk’s ‘R Us</td>
<td>Kettering, OH</td>
</tr>
</tbody>
</table>
This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.