Chapter 6

Database Design 2: Design Method
Objectives

• Discuss the general process and goals of database design
• Define user views and explain their function
• Define Database Design Language (DBDL) and use it to document database designs
• Create an entity-relationship (E-R) diagram to visually represent a database design
• Present a method for database design at the information level and view examples illustrating this method
Objectives (continued)

• Explain the physical-level design process
• Discuss top-down and bottom-up approaches to database design and examine the advantages and disadvantages of both methods
• Use a survey form to obtain information from users prior to beginning the database design process
• Review existing documents to obtain information prior to beginning the database design
Objectives (continued)

• Discuss special issues related to implementing one-to-one relationships and many-to-many relationships involving more than two entities
• Discuss entity subtypes and their relationships to nulls
• Learn how to avoid potential problems when merging third normal form relations
• Examine the entity-relationship model for representing and designing databases
Introduction

• Two-step process for database design
• **Information-level design**: completed *independently* of any particular DBMS
• **Physical-level design**: information-level design adapted for the specific DBMS that will be used
  – Must consider characteristics of the particular DBMS
User Views

• **User view**: set of requirements necessary to support operations of a particular database user

• **Cumulative design**: supports all user views encountered during design process
Information-Level Design Method

• For each user view:
  1. Represent the user view as a collection of tables
  2. Normalize these tables
  3. Identify all keys in these tables
  4. Merge the result of Steps 1 through 3 into the cumulative design
Represent the User View As a Collection of Tables

• Step 1: Determine the entities involved and create a separate table for each type of entity
• Step 2: Determine the primary key for each table
• Step 3: Determine the properties for each entity
• Step 4: Determine relationships between the entities
  – One-to-many
  – Many-to-many
  – One-to-one
Represent the User View As a Collection of Tables (continued)

- One-to-many relationship: include primary key of the “one” table as a foreign key in the “many” table
- **Many-to-many relationship**: create a new table whose primary key is the combination of the primary keys of the original tables
- **One-to-one relationship**: simplest implementation is to treat it as a one-to-many relationship
Normalize the Tables

• Normalize each table
• Target is third normal form
  – Careful planning in early phases of the process usually rules out need to consider fourth normal form
Identify All Keys

• For each table, identify:
  – Primary key
  – Alternate keys
  – Secondary keys
  – Foreign keys

• Alternate key: column(s) that could have been chosen as a primary key but was not

• **Secondary keys**: columns of interest strictly for retrieval purposes
Identify All Keys (continued)

• Foreign key: column(s) in one table that is required to match value of the primary key for some row in another table or is required to be null
  – Used to create relationships between tables
  – Used to enforce certain types of integrity constraints
Types of Primary Keys

- **Natural key**: consists of a column that uniquely identifies an entity
  - Also called a *logical key* or an *intelligent key*
- **Artificial key**: column created for an entity to serve solely as the primary key and that is visible to users
- **Surrogate key**: system-generated; usually hidden from users
  - Also called a *synthetic key*
Database Design Language (DBDL)

- Table name followed by columns in parentheses
  - Primary key column(s) underlined
- AK identifies alternate keys
- SK identifies secondary keys
- FK identifies foreign keys
  - Foreign keys followed by an arrow pointing to the table identified by the foreign key
Database Design Language (DBDL) (continued)

FIGURE 6-1: DBDL for the Employee table

```
Employee (EmployeeNum, LastName, FirstName, Street, City, State, Zip, 
WageRate, SocSecNum, DepartmentNum)
AK SocSecNum
SK LastName
FK DepartmentNum → Department
```
Entity-Relationship (E-R) Diagrams

• Visually represents database structure
• Rectangle represents each entity
  – Entity’s name appears above the rectangle
• Primary key for each entity appears above the line in the entity’s rectangle
• Other columns of entity appear below the line in rectangle
Entity-Relationship (E-R) Diagrams (continued)

- Letters AK, SK, and FK appear in parentheses following the alternate key, secondary key, and foreign key, respectively.
- For each foreign key, a line leads from the rectangle for the table being identified to the rectangle for the table containing the foreign key.
- Text uses IDEF1X style of E-R diagram.
Entity-Relationship (E-R) Diagrams (continued)

FIGURE 6-2: E-R diagram
Merge the Result into the Design

• Combine tables that have the same primary key to form a new table

• New table:
  – Primary key is same as the primary key in the tables combined
  – Contains all the columns from the tables combined
  – If duplicate columns, remove all but one copy of the column

• Make sure new design is in third normal form
Merge the Result into the Design (continued)

FIGURE 6-3: Information-level design method

1. Represent the user view as a collection of tables (relations).
2. Normalize these tables.
3. Represent all keys.
4. Merge the result of the previous steps into the cumulative design.
Database Design Examples

• Develop an information-level design
• Company stores information about sales reps, customers, parts, and orders
• User view requirements
• Constraints

```
Rep (RepNum, LastName, FirstName, Street, City, State, Zip, Commission, Rate)
```

FIGURE 6-4: Cumulative design after first user view
Database Design Examples (continued)

FIGURE 6-6: Cumulative design after third user view
Database Design Examples (continued)

FIGURE 6-8: Final information-level design
Database Design Examples (continued)

- Henry Books database: information about branches, publishers, authors, and books
- User view requirements

```
Publisher (PublisherCode, PublisherName, City)
SK PublisherName
```

FIGURE 6-9: DBDL for Book database after first user view
Database Design Examples (continued)

**FIGURE 6-10: DBDL for Book database after second user view**

<table>
<thead>
<tr>
<th>Publisher (PublisherCode, PublisherName, City)</th>
<th>SK PublisherName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch (BranchNum, BranchName, BranchLocation)</td>
<td>SK BranchName</td>
</tr>
</tbody>
</table>
FIGURE 6-13: Cumulative design after fifth user view
Physical-Level Design

• Undertaken after information-level design completion
• Most DBMSs support primary, candidate, secondary, and foreign keys
• To enforce restrictions, DB programmers must include logic in their programs
Top-Down Versus Bottom-Up

• **Bottom-up design method**
  – Design starts at low level
  – Specific user requirements drive design process

• **Top-down design method**
  – Begins with general database that models overall enterprise
  – Refines model until design supports all necessary applications
Survey Form

• Used to collect information from users
• Must contain particular elements
  – Entity information
  – Attribute (column) information
  – Relationships
  – Functional dependencies
  – Processing information
Obtaining Information from Existing Documents

- Existing documents can furnish information about database design
- Identify and list all columns and give them appropriate names
- Identify functional dependencies
- Determine the tables and assign columns
Obtaining Information from Existing Documents (continued)

FIGURE 6-14: Invoice for Holt Distributors
Obtaining Information from Existing Documents (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvoiceNumber</td>
</tr>
<tr>
<td>InvoiceDate</td>
</tr>
<tr>
<td>CustomerNumber</td>
</tr>
<tr>
<td>CustomerSoldToName</td>
</tr>
<tr>
<td>CustomerSoldToAddressLine1</td>
</tr>
<tr>
<td>CustomerSoldToAddressLine2</td>
</tr>
<tr>
<td>CustomerSoldToCity</td>
</tr>
<tr>
<td>CustomerSoldToState</td>
</tr>
<tr>
<td>CustomerSoldToZip</td>
</tr>
<tr>
<td>CustomerShipToName</td>
</tr>
<tr>
<td>CustomerShipToAddress</td>
</tr>
<tr>
<td>CustomerShipToCity</td>
</tr>
<tr>
<td>CustomerShipToState</td>
</tr>
<tr>
<td>CustomerShipToZip</td>
</tr>
<tr>
<td>CustomerPONumber</td>
</tr>
<tr>
<td>OrderNumber</td>
</tr>
<tr>
<td>OrderDate</td>
</tr>
<tr>
<td>ShipDate</td>
</tr>
<tr>
<td>CustomerRepNumber</td>
</tr>
<tr>
<td>CustomerRepLastName</td>
</tr>
<tr>
<td>CustomerRepFirstName</td>
</tr>
<tr>
<td>ItemNumber</td>
</tr>
<tr>
<td>ItemDescription</td>
</tr>
<tr>
<td>ItemQuantityOrdered</td>
</tr>
<tr>
<td>ItemQuantityShipped</td>
</tr>
<tr>
<td>ItemQuantityBackordered</td>
</tr>
<tr>
<td>ItemPrice</td>
</tr>
<tr>
<td>ItemAmount</td>
</tr>
<tr>
<td>Freight</td>
</tr>
<tr>
<td>InvoiceTotal</td>
</tr>
</tbody>
</table>

FIGURE 6-15: List of possible attributes for the Holt Distributors invoice
Obtaining Information from Existing Documents (continued)

<table>
<thead>
<tr>
<th>Function Dependency</th>
<th>Attribute(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerNumber</td>
<td>CustomerSoldToName</td>
</tr>
<tr>
<td></td>
<td>CustomerSoldToAddressLine1</td>
</tr>
<tr>
<td></td>
<td>CustomerSoldToAddressLine2</td>
</tr>
<tr>
<td></td>
<td>CustomerSoldToCity</td>
</tr>
<tr>
<td></td>
<td>CustomerSoldToState</td>
</tr>
<tr>
<td></td>
<td>CustomerSoldToZip</td>
</tr>
<tr>
<td></td>
<td>CustomerRepNumber</td>
</tr>
<tr>
<td></td>
<td>CustomerRepLastName</td>
</tr>
<tr>
<td></td>
<td>CustomerRepFirstName</td>
</tr>
<tr>
<td>ItemNumber</td>
<td>ItemDescription</td>
</tr>
<tr>
<td></td>
<td>ItemPrice</td>
</tr>
<tr>
<td>InvoiceNumber</td>
<td>InvoiceDate</td>
</tr>
<tr>
<td></td>
<td>OrderNumber</td>
</tr>
<tr>
<td></td>
<td>ShipDate</td>
</tr>
<tr>
<td></td>
<td>Freight</td>
</tr>
<tr>
<td></td>
<td>InvoiceTotal</td>
</tr>
<tr>
<td>OrderNumber</td>
<td>OrderDate</td>
</tr>
<tr>
<td></td>
<td>CustomerPONumber</td>
</tr>
<tr>
<td></td>
<td>CustomerShipToName</td>
</tr>
<tr>
<td></td>
<td>CustomerShipToAddressLine1</td>
</tr>
<tr>
<td></td>
<td>CustomerShipToAddressLine2</td>
</tr>
<tr>
<td></td>
<td>CustomerShipToCity</td>
</tr>
<tr>
<td></td>
<td>CustomerShipToState</td>
</tr>
<tr>
<td></td>
<td>CustomerShipToZip</td>
</tr>
<tr>
<td>OrderNumber, ItemNumber</td>
<td>ItemQuantityOrdered (added when order is entered)</td>
</tr>
<tr>
<td></td>
<td>ItemQuantityShipped (added during invoicing)</td>
</tr>
<tr>
<td></td>
<td>ItemQuantityBackordered (added during invoicing)</td>
</tr>
<tr>
<td></td>
<td>ItemPrice (added when order is entered)</td>
</tr>
</tbody>
</table>

FIGURE 6-17: Revised list of functional dependencies for the Holt Distributors invoice
Obtaining Information from Existing Documents (continued)

FIGURE 6-19: Expanded list of entities

Invoice
Customer
Rep
Part
Orders
OrderLine
One-to-One Relationship Considerations

• Simply include the primary key of each table as a foreign key in the other table
  – No guarantee that the information will match

• One solution: create a single table
  – Workable, but not the best solution

• Better solution
  – Create separate tables for customers and sales reps
  – Include the primary key of one of them as a foreign key in the other
One-to-One Relationship Considerations (continued)

**Solution 1:**

<table>
<thead>
<tr>
<th>RepNum</th>
<th>LastName</th>
<th>FirstName</th>
<th>CustomerNum</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Kaiser</td>
<td>Valerie</td>
<td>148</td>
</tr>
<tr>
<td>35</td>
<td>Hull</td>
<td>Richard</td>
<td>282</td>
</tr>
<tr>
<td>65</td>
<td>Perez</td>
<td>Juan</td>
<td>356</td>
</tr>
</tbody>
</table>

**Solution 2:**

<table>
<thead>
<tr>
<th>RepNum</th>
<th>LastName</th>
<th>FirstName</th>
<th>CustomerNum</th>
<th>CustomerName</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Kaiser</td>
<td>Valerie</td>
<td>148</td>
<td>Al’s Appliance and Sport</td>
</tr>
<tr>
<td>35</td>
<td>Hull</td>
<td>Richard</td>
<td>282</td>
<td>Brookings Direct</td>
</tr>
<tr>
<td>65</td>
<td>Perez</td>
<td>Juan</td>
<td>356</td>
<td>Ferguson’s</td>
</tr>
</tbody>
</table>

**FIGURE 6-23:** One-to-one relationship implemented by including the primary key of one table as the foreign key (and alternate key) in the other table
Many-to-Many Relationship Considerations

• Complex issues arise when more than two entities are related in a many-to-many relationship

• **Many-to-many-to-many relationship**: involves multiple entities

• Deciding between a single many-to-many-to-many relationship and two (or three) many-to-many relationships
  – Crucial issue: independence
Many-to-Many Relationship Considerations (continued)

FIGURE 6-25: Result obtained by splitting the Sales table into three tables
Many-to-Many Relationship Considerations (continued)

FIGURE 6-26: Result obtained by joining three tables—the second and third rows are in error!
Nulls and Entity Subtypes

• Null
  – Special value
  – Represents *absence* of a value in a field
  – Used when a value is unknown or inapplicable

• Splitting tables to avoid use of null values

• **Entity subtype**: table that is a subtype of another table
Nulls and Entity Subtypes (continued)

FIGURE 6-27: Student table split to avoid use of null values
Nulls and Entity Subtypes (continued)

- Subtype called a **category** in IDEF1X terminology
- **Incomplete category**: records that do not fall into the subtype
- **Complete categories**: all records fall into the categories
Nulls and Entity Subtypes (continued)

FIGURE 6-29: Entity subtype in an E-R diagram
FIGURE 6-32: Two entity subtypes—incomplete categories
Nulls and Entity Subtypes (continued)

FIGURE 6-33: Two entity subtypes—complete categories
Avoiding Problems with Third Normal Form When Merging Tables

• When combining third normal form tables, the result might not be in third normal form
• Be cautious when representing user views
• Always attempt to determine whether determinants exist and include them in tables
The Entity-Relationship Model

- An approach to representing data in a database
- Entities are drawn as rectangles
- Relationships are drawn as diamonds with lines connecting the entities involved in relationships
- **Composite entity:** exists to implement a many-to-many relationship
- **Existence dependency:** existence of one entity depends on the existence of another related entity
- **Weak entity:** depends on another entity for its own existence
The Entity-Relationship Model (continued)

FIGURE 6-34: One-to-many relationship
FIGURE 6-35: Many-to-many relationship
The Entity-Relationship Model (continued)

FIGURE 6-36: Many-to-many-to-many relationship
The Entity-Relationship Model (continued)

FIGURE 6-37: One-to-many relationship with attributes added
The Entity-Relationship Model (continued)

FIGURE 6-38: Many-to-many relationship with attributes
The Entity-Relationship Model (continued)

FIGURE 6-39: Composite entity
The Entity-Relationship Model (continued)

FIGURE 6-40: Complete E-R diagram for the Premiere Products database
The Entity-Relationship Model (continued)

FIGURE 6-41: E-R diagram with an existence dependency and a weak entity
The Entity-Relationship Model (continued)

• **Cardinality**: number of items that must be included in a relationship
  – An entity in a relationship with minimum cardinality of zero plays an *optional role* in the relationship
  – An entity with a minimum cardinality of one plays a *mandatory role* in the relationship
The Entity-Relationship Model (continued)

FIGURE 6-43: E-R diagram that represents cardinality
Summary

• Database design is a two-part process: information-level design (not dependent on a particular DBMS) and physical-level design (appropriate for the particular DBMS being used)

• User view: set of necessary requirements to support a particular user’s operations

• Information-level design steps for each user view: represent the user view as a collection of tables, normalize these tables, represent all keys (primary, alternate, secondary, and foreign), and merge the results into the cumulative design
Summary (continued)

• Database design is represented in Database Design Language (DBDL)

• Designs can be represented visually using entity-relationship (E-R) diagrams

• Physical-level design process consists of creating a table for each entity in the DBDL design

• Design method presented in this chapter is bottom-up

• Survey form is useful for documenting the information gathered for database design process
Summary (continued)

• To obtain information from existing documents, list all attributes present in the documents, identify potential functional dependencies, make a tentative list of tables, and use the functional dependencies to refine the list.

• To implement a one-to-one relationship, include primary key of one table in the other table as a foreign key and indicate the foreign key as an alternate key.
Summary (continued)

• If a table’s primary key consists of three (or more) columns, determine whether there are independent relationships between pairs of these columns

• If a table contains columns that can be null and the nulls mean that the column is inapplicable for some rows, you can split the table, placing the null column(s) in separate tables

• The result of merging third normal form tables may not be in third normal form

• Entity-relationship (E-R) model represents the structure of a database using an E-R diagram