Concepts of Database Management Seventh Edition

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

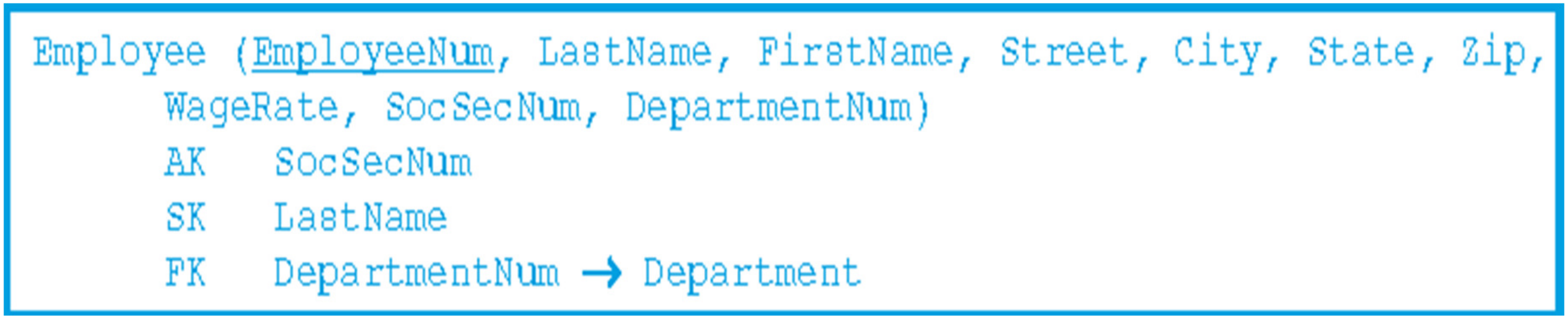
Types of Primary Keys

- **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)



```
Employee (EmployeeNum, LastName, FirstName, Street, City, State, Zip,
      WageRate, SocSecNum, DepartmentNum)
      AK    SocSecNum
      SK    LastName
      FK    DepartmentNum → Department
```

FIGURE 6-1: DBDL for the Employee table

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)

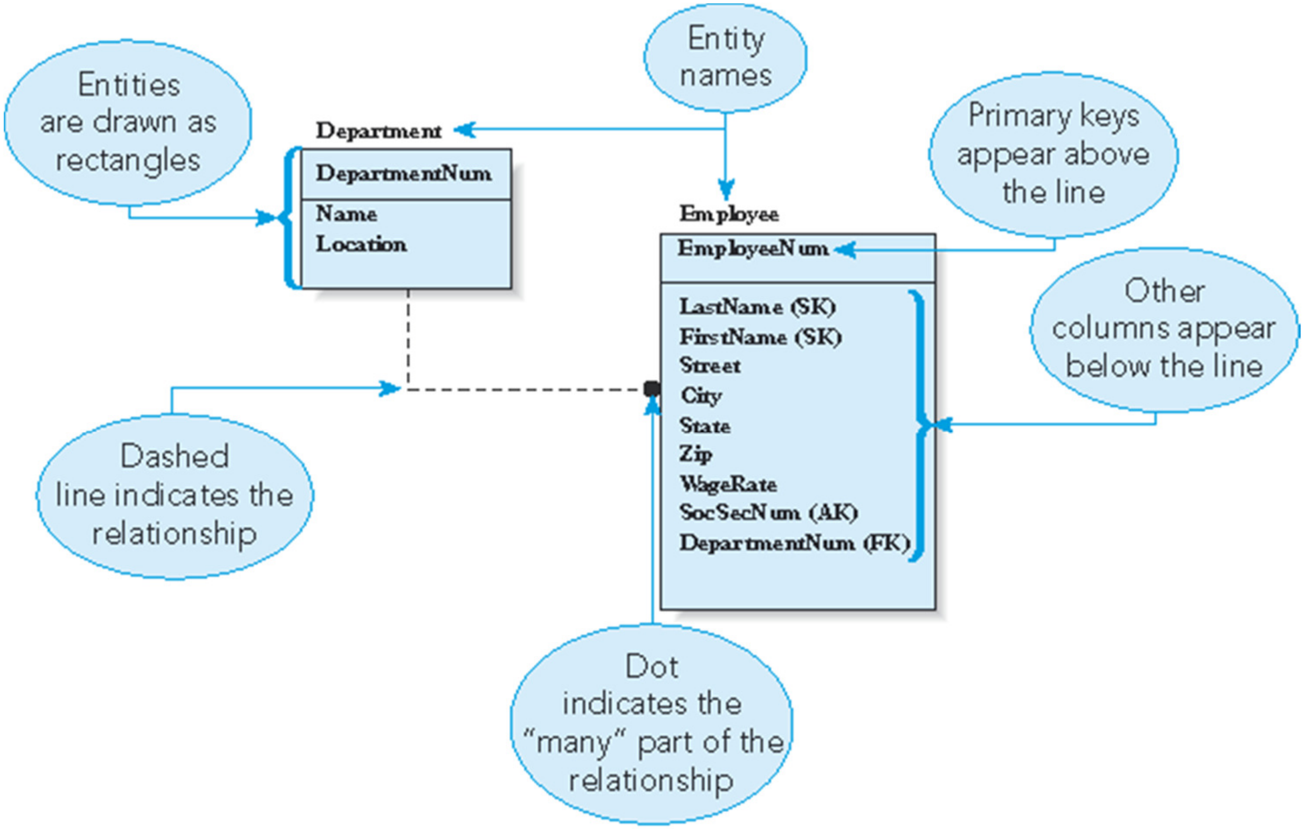

Entities are drawn as rectangles
Entity names
Primary keys appear above the line
Other columns appear below the line
Dashed line indicates the relationship
Dot indicates the "many" part of the relationship
Department
DepartmentNum
Name
Location
Employee
EmployeeNum
LastName (SK)
FirstName (SK)
Street
City
State
Zip
WageRate
SocSecNum (AK)
DepartmentNum (FK)

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)

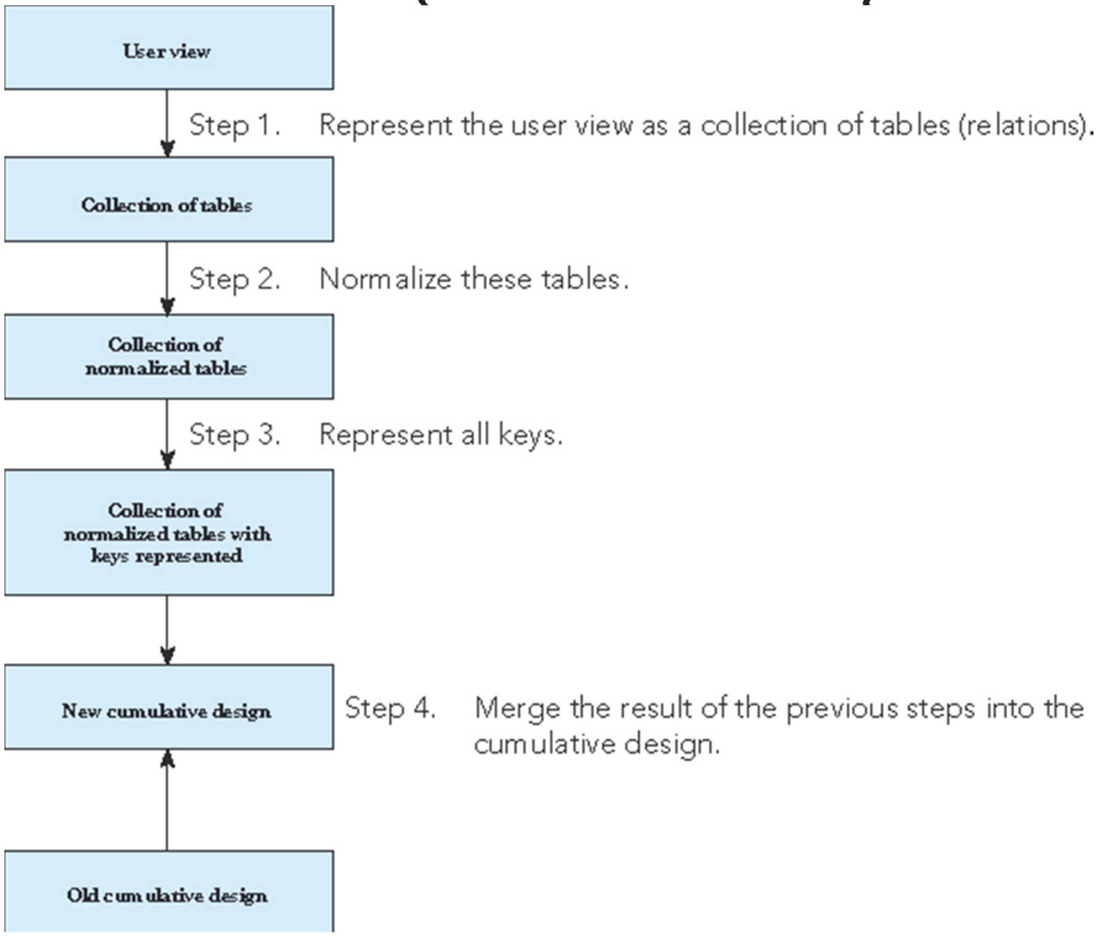
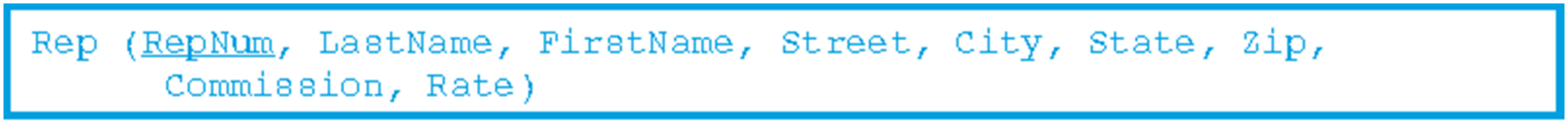

User view
Step 1. Represent the user view as a collection of tables (relations).
Collection of tables
Step 2. Normalize these tables.
Collection of normalized tables
Step 3. Represent all keys.
Collection of normalized tables with keys represented
New cumulative design
Step 4. Merge the result of the previous steps into the cumulative design.
Old cumulative design

FIGURE 6-3: Information-level design method

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)

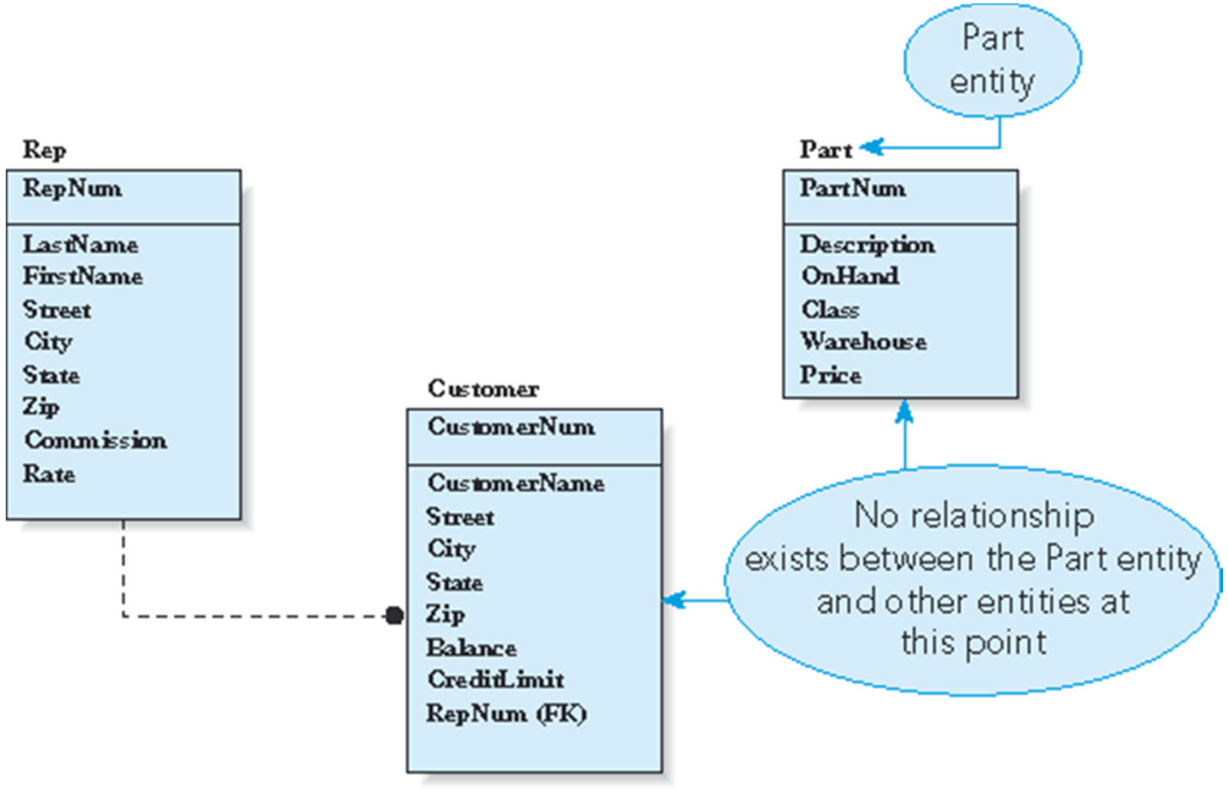


FIGURE 6-6: Cumulative design after third user view

Database Design Examples (continued)

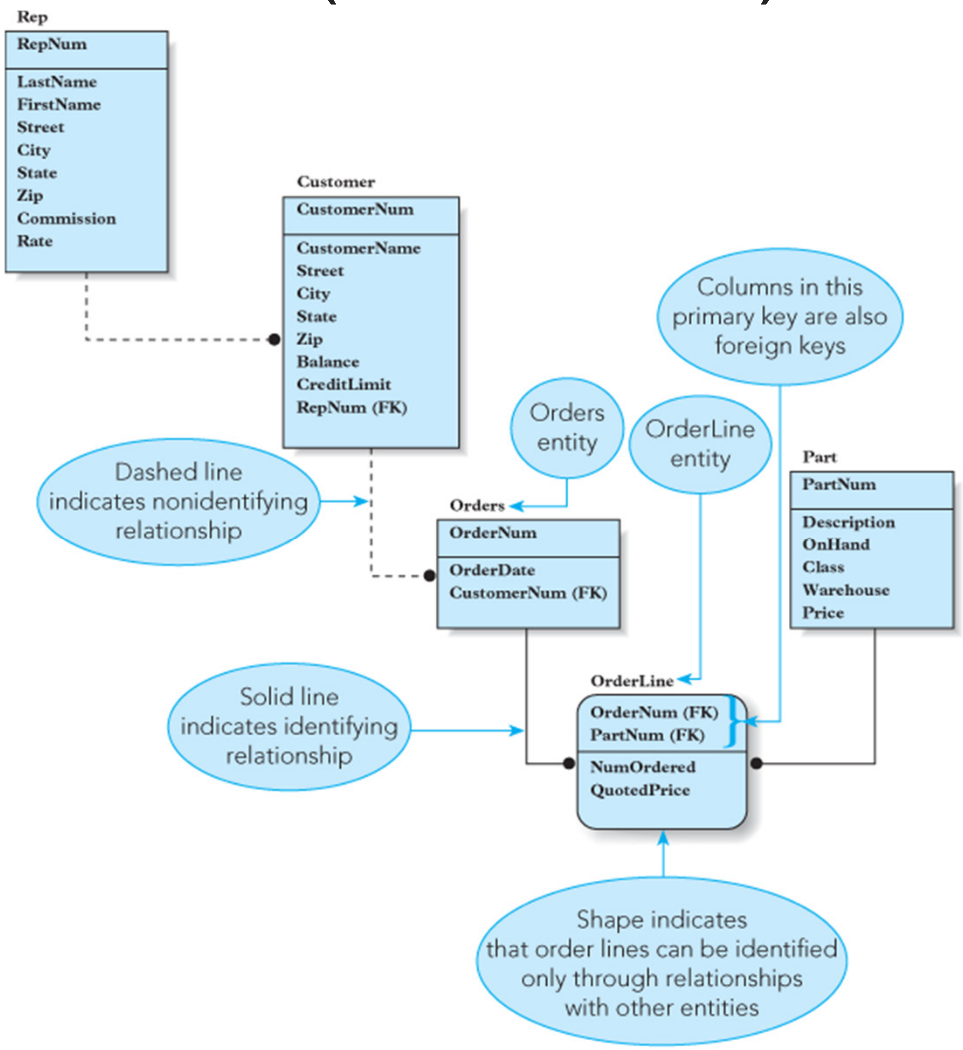
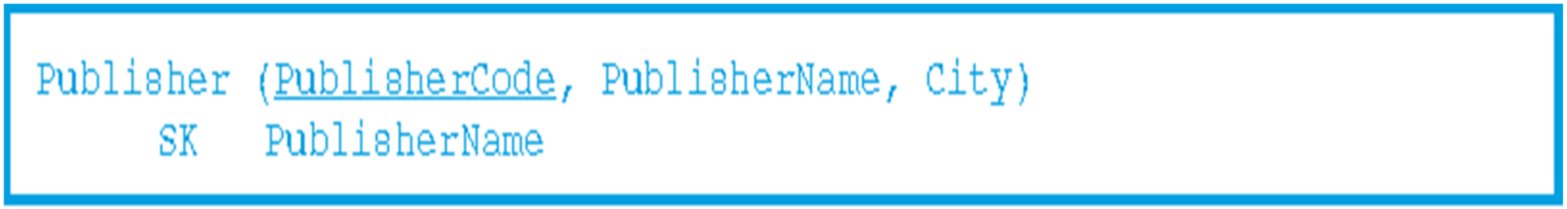


FIGURE 6-8: Final information-level design

Database Design Examples (continued)

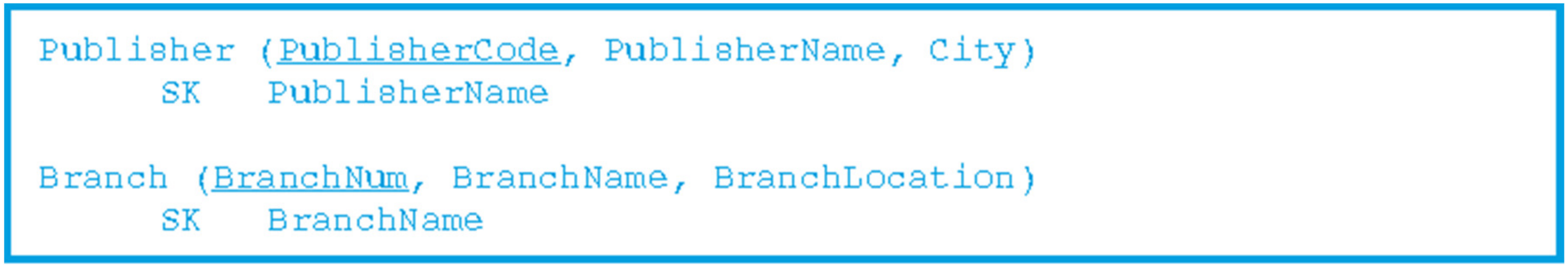
- Henry Books database: information about branches, publishers, authors, and books
- User view requirements



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

FIGURE 6-9: DBDL for Book database after first user view

Database Design Examples (continued)



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

Branch (BranchNum, BranchName, BranchLocation)
      SK   BranchName
```

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

Database Design Examples (continued)

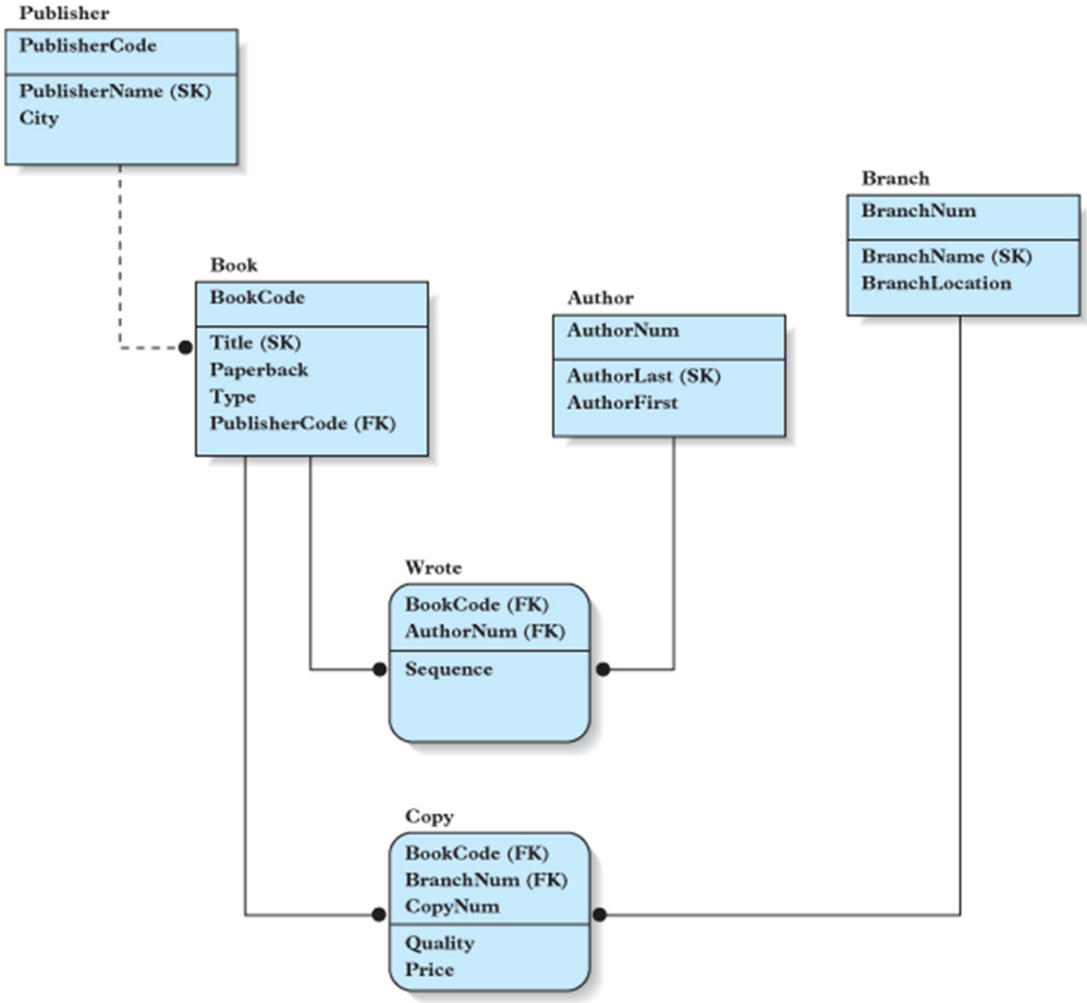


FIGURE 6-13: Cumulative design after fifth user view

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

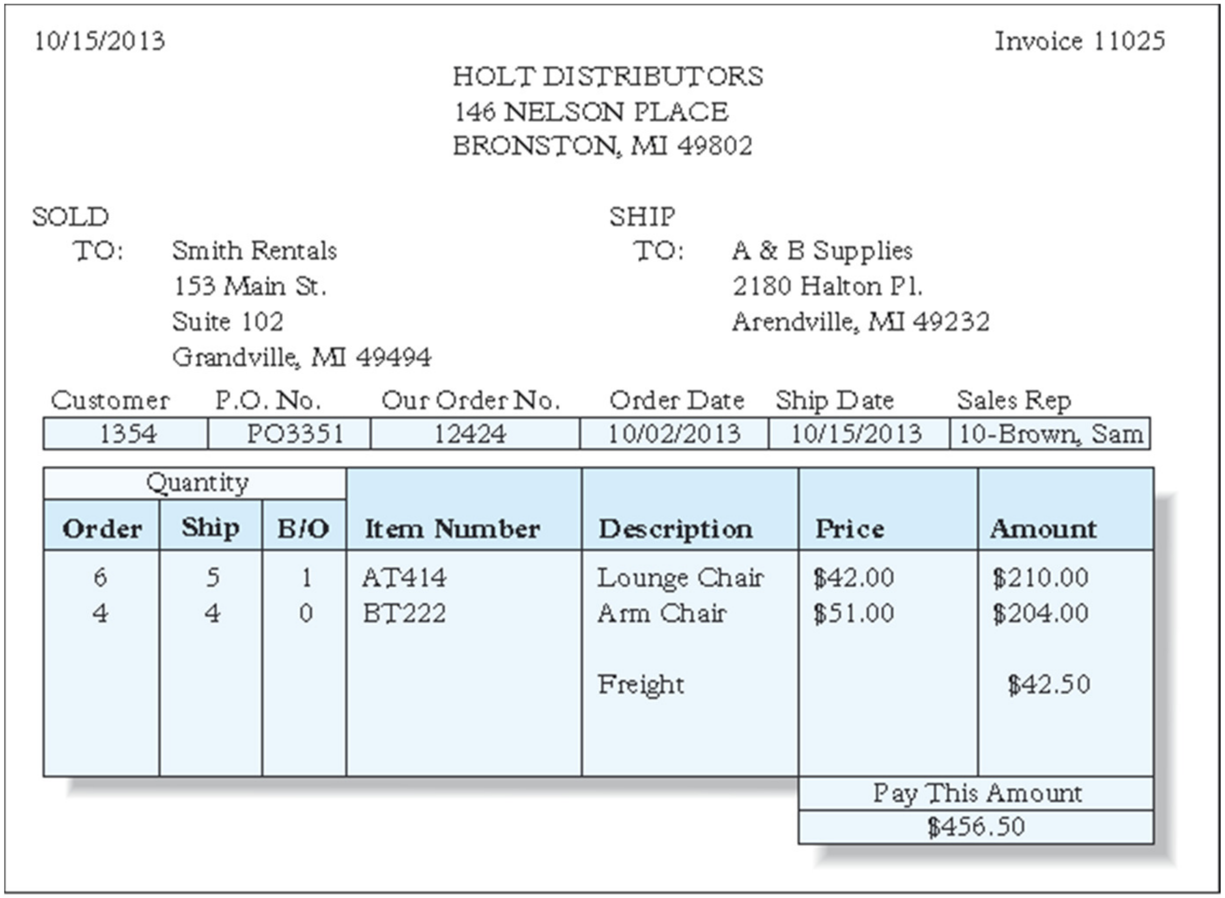
Survey Form

- 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)



10/15/2013

Invoice 11025

HOLT DISTRIBUTORS
146 NELSON PLACE
BRONSTON, MI 49802

SOLD
TO: Smith Rentals
153 Main St.
Suite 102
Grandville, MI 49494

SHIP
TO: A & B Supplies
2180 Halton Pl.
Arendville, MI 49232

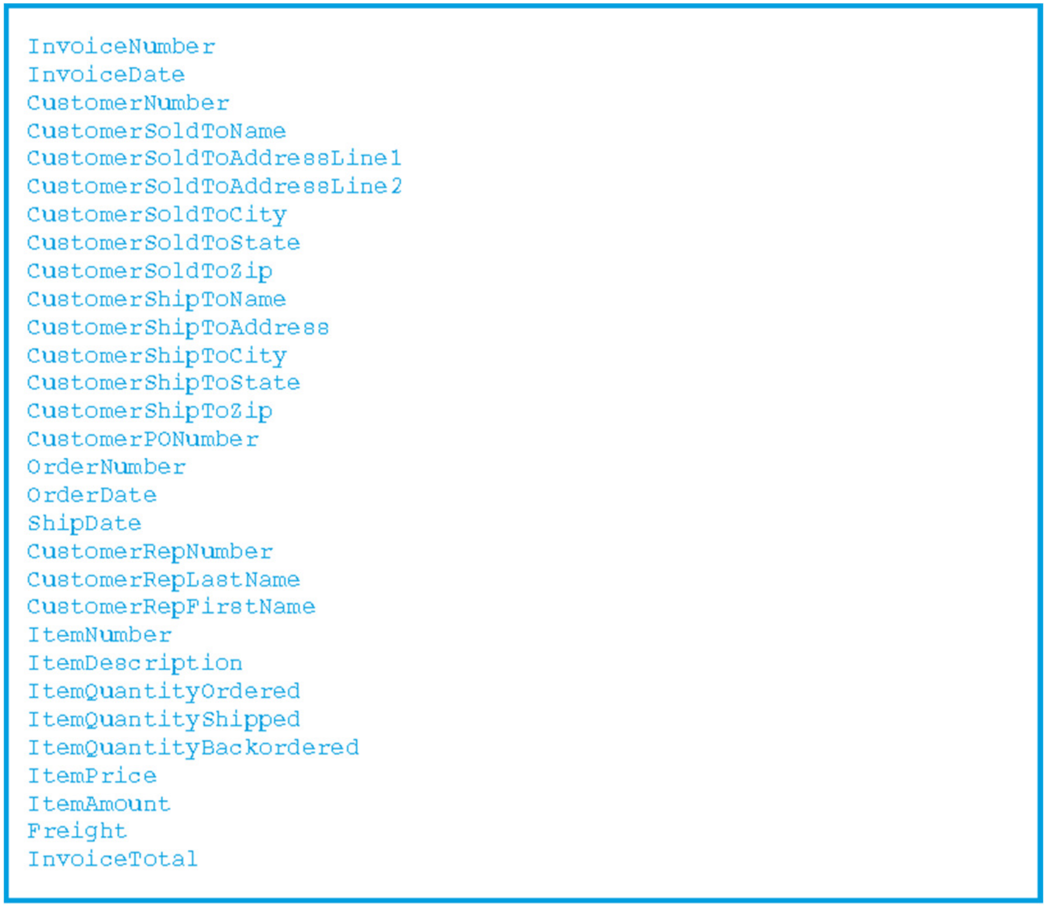
Customer	P.O. No.	Our Order No.	Order Date	Ship Date	Sales Rep
1354	PO3351	12424	10/02/2013	10/15/2013	10-Brown, Sam

Quantity						
Order	**Ship**	**B/O**	**Item Number**	**Description**	**Price**	**Amount**
6	5	1	AT414	Lounge Chair	$42.00	$210.00
4	4	0	BT222	Arm Chair	$51.00	$204.00
				Freight		$42.50

Pay This Amount
$456.50

FIGURE 6-14: Invoice for Holt Distributors

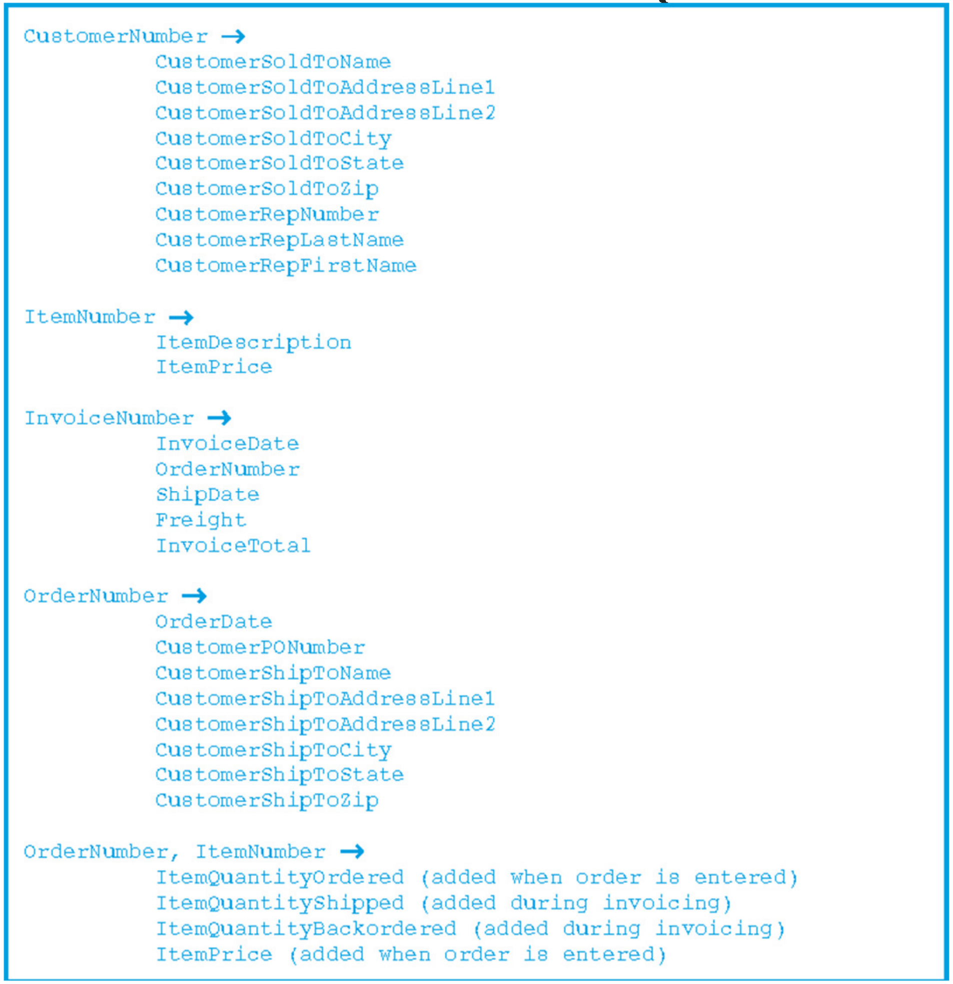
Obtaining Information from Existing Documents (continued)



```
InvoiceNumber
InvoiceDate
CustomerNumber
CustomerSoldToName
CustomerSoldToAddressLine1
CustomerSoldToAddressLine2
CustomerSoldToCity
CustomerSoldToState
CustomerSoldToZip
CustomerShipToName
CustomerShipToAddress
CustomerShipToCity
CustomerShipToState
CustomerShipToZip
CustomerPONumber
OrderNumber
OrderDate
ShipDate
CustomerRepNumber
CustomerRepLastName
CustomerRepFirstName
ItemNumber
ItemDescription
ItemQuantityOrdered
ItemQuantityShipped
ItemQuantityBackordered
ItemPrice
ItemAmount
Freight
InvoiceTotal
```

FIGURE 6-15: List of possible attributes for the Holt Distributors invoice

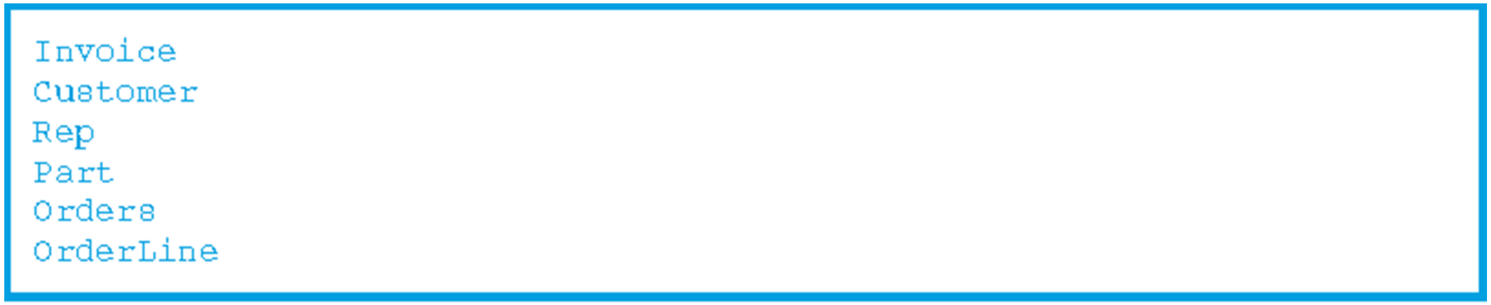
Obtaining Information from Existing Documents (continued)



```
CustomerNumber →
          CustomerSoldToName
          CustomerSoldToAddressLine1
          CustomerSoldToAddressLine2
          CustomerSoldToCity
          CustomerSoldToState
          CustomerSoldToZip
          CustomerRepNumber
          CustomerRepLastName
          CustomerRepFirstName

ItemNumber →
          ItemDescription
          ItemPrice

InvoiceNumber →
          InvoiceDate
          OrderNumber
          ShipDate
          Freight
          InvoiceTotal

OrderNumber →
          OrderDate
          CustomerPONumber
          CustomerShipToName
          CustomerShipToAddressLine1
          CustomerShipToAddressLine2
          CustomerShipToCity
          CustomerShipToState
          CustomerShipToZip

OrderNumber, ItemNumber →
          ItemQuantityOrdered (added when order is entered)
          ItemQuantityShipped (added during invoicing)
          ItemQuantityBackordered (added during invoicing)
          ItemPrice (added when order is entered)
```

FIGURE 6-17: Revised list of functional dependencies for the Holt Distributors invoice

Obtaining Information from Existing Documents (continued)



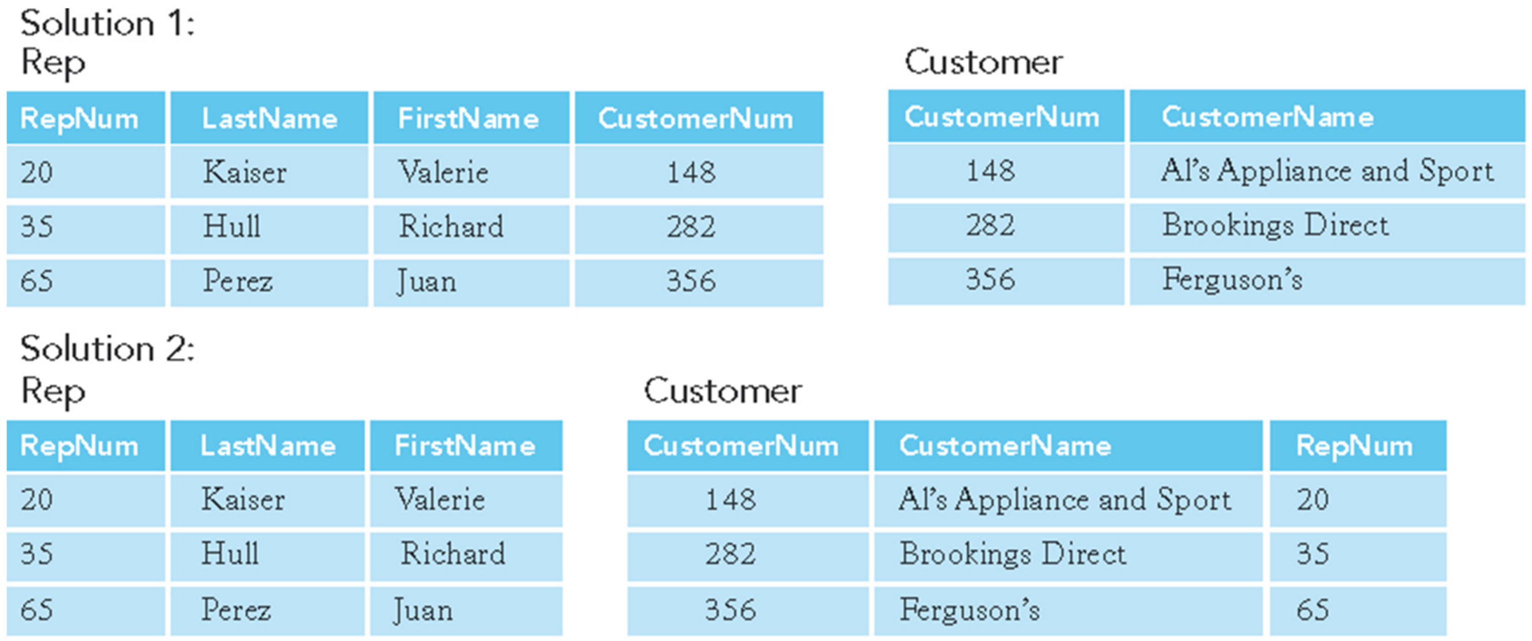
```
Invoice
Customer
Rep
Part
Orders
OrderLine
```

FIGURE 6-19: Expanded list of entities

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:

Rep

RepNum	LastName	FirstName	CustomerNum
20	Kaiser	Valerie	148
35	Hull	Richard	282
65	Perez	Juan	356

Customer

CustomerNum	CustomerName
148	Al's Appliance and Sport
282	Brookings Direct
356	Ferguson's

Solution 2:

Rep

RepNum	LastName	FirstName
20	Kaiser	Valerie
35	Hull	Richard
65	Perez	Juan

Customer

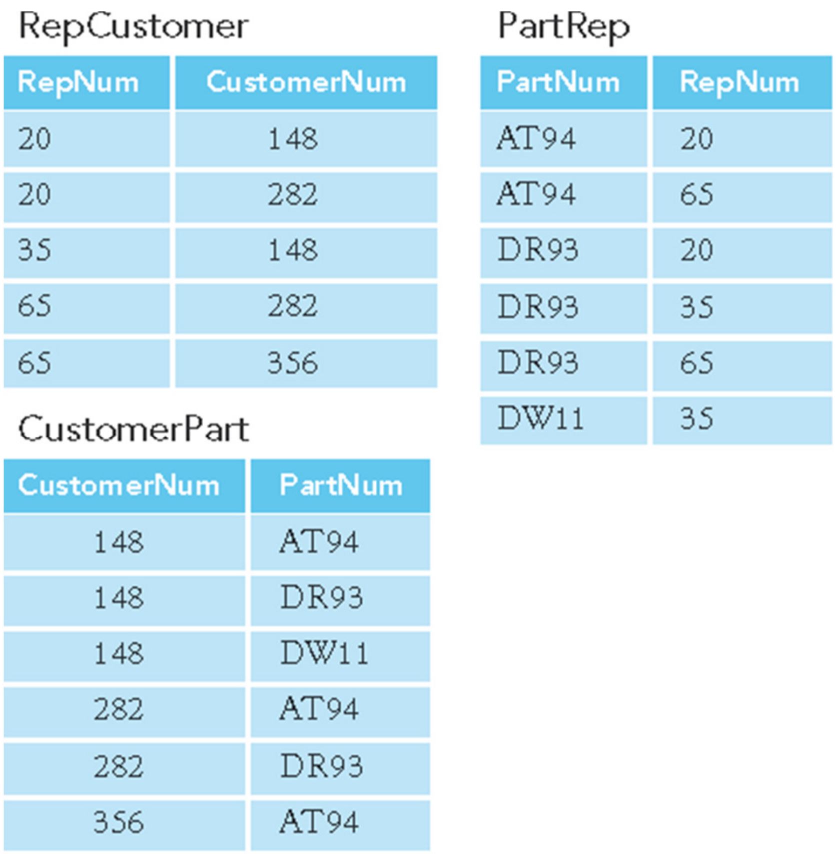
CustomerNum	CustomerName	RepNum
148	Al's Appliance and Sport	20
282	Brookings Direct	35
356	Ferguson's	65

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)



RepCustomer

RepNum	CustomerNum
20	148
20	282
35	148
65	282
65	356

PartRep

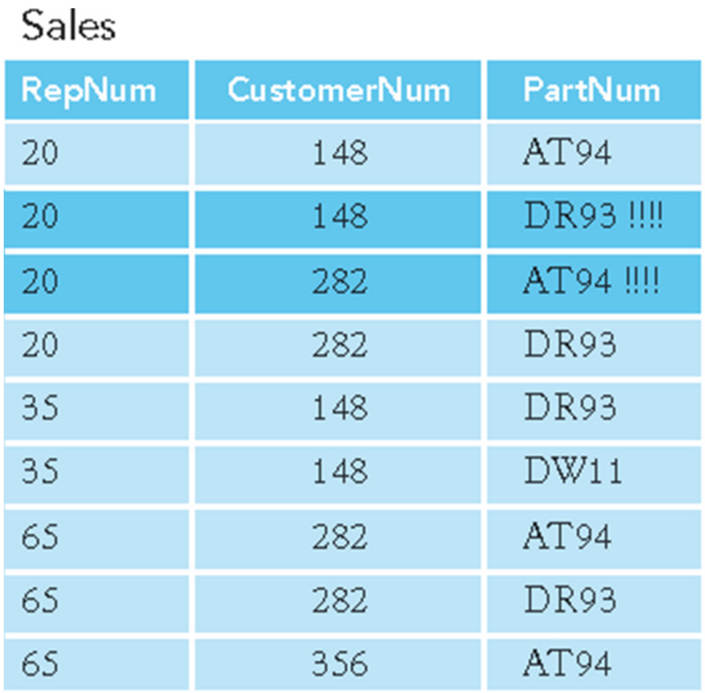
PartNum	RepNum
AT94	20
AT94	65
DR93	20
DR93	35
DR93	65
DW11	35

CustomerPart

CustomerNum	PartNum
148	AT94
148	DR93
148	DW11
282	AT94
282	DR93
356	AT94

FIGURE 6-25: Result obtained by splitting the Sales table into three tables

Many-to-Many Relationship Considerations (continued)



Sales

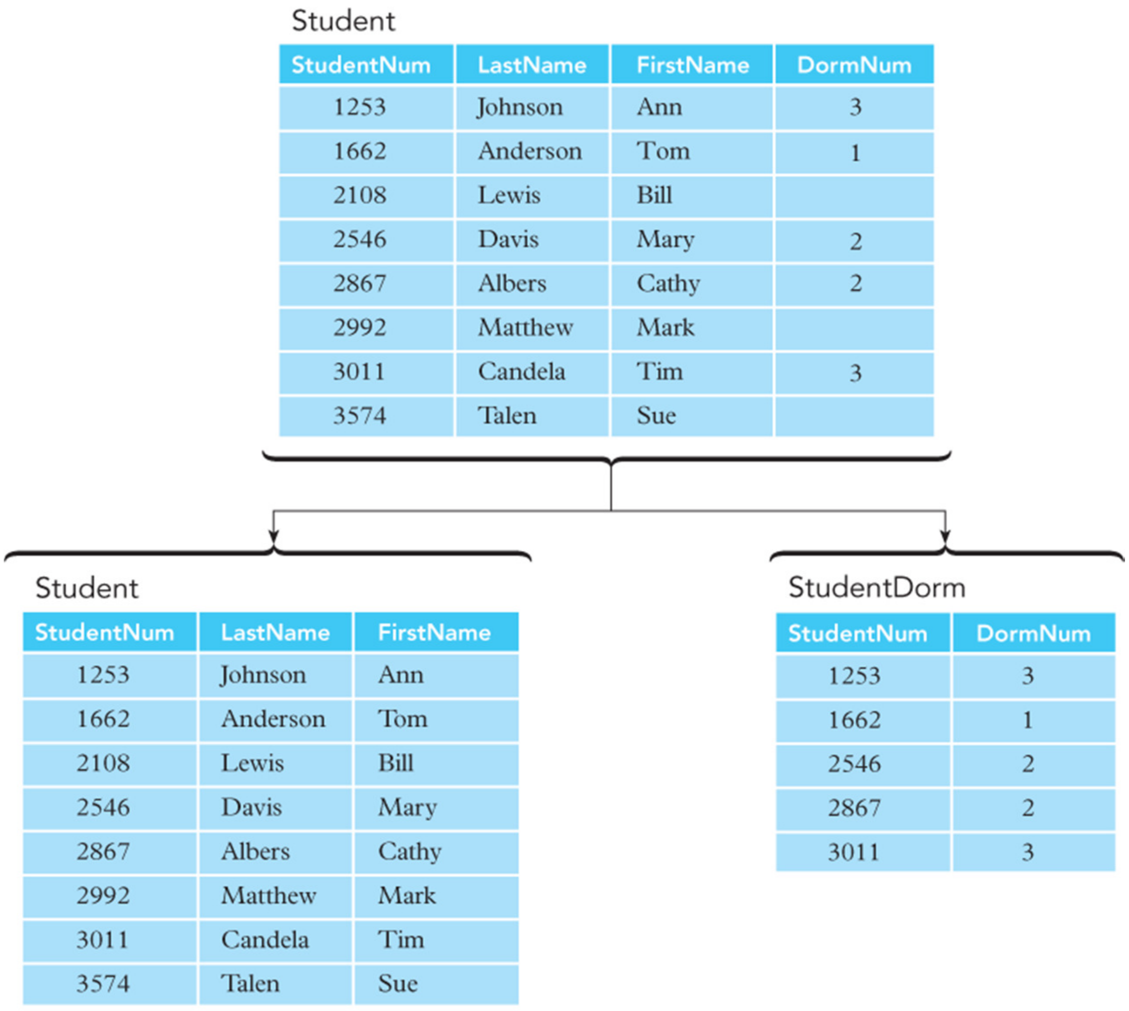
RepNum	CustomerNum	PartNum
20	148	AT94
20	148	DR93 !!!!
20	282	AT94 !!!!
20	282	DR93
35	148	DR93
35	148	DW11
65	282	AT94
65	282	DR93
65	356	AT94

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)



Student

StudentNum	LastName	FirstName	DormNum
1253	Johnson	Ann	3
1662	Anderson	Tom	1
2108	Lewis	Bill	
2546	Davis	Mary	2
2867	Albers	Cathy	2
2992	Matthew	Mark	
3011	Candela	Tim	3
3574	Talen	Sue	

Student

StudentNum	LastName	FirstName
1253	Johnson	Ann
1662	Anderson	Tom
2108	Lewis	Bill
2546	Davis	Mary
2867	Albers	Cathy
2992	Matthew	Mark
3011	Candela	Tim
3574	Talen	Sue

StudentDorm

StudentNum	DormNum
1253	3
1662	1
2546	2
2867	2
3011	3

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)

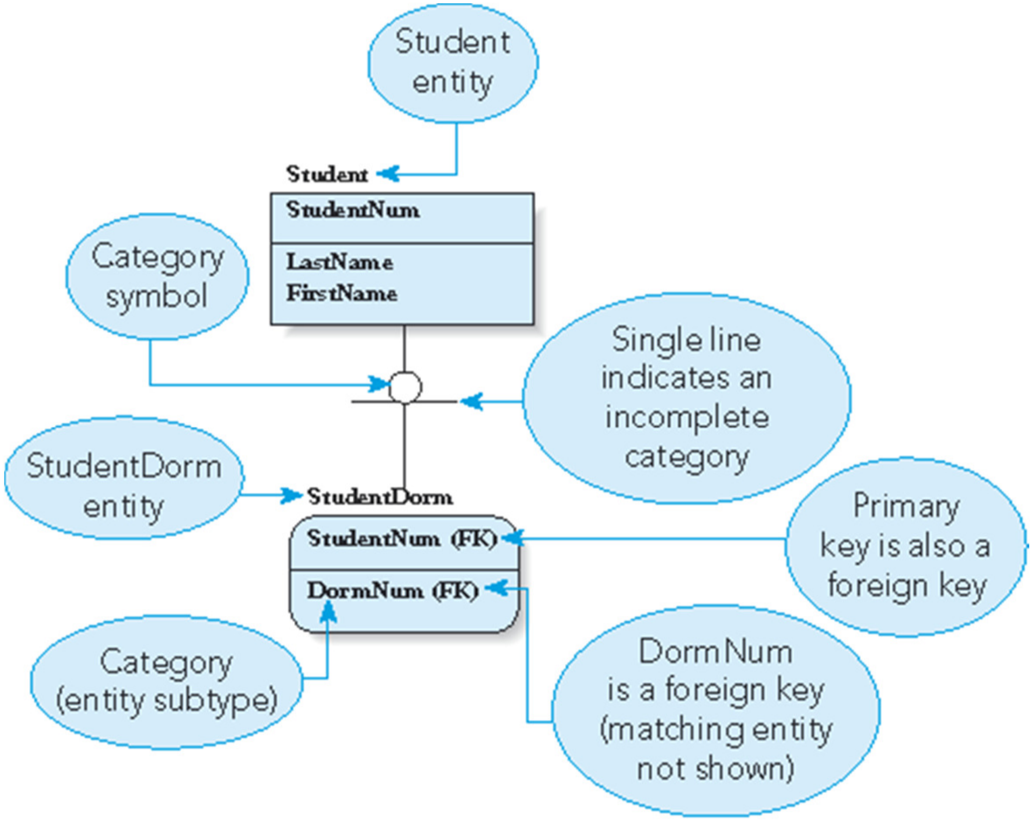

Student entity
Student
StudentNum
LastName
FirstName
Category symbol
Single line indicates an incomplete category
StudentDorm entity
StudentDorm
StudentNum (FK)
DormNum (FK)
Primary key is also a foreign key
Category (entity subtype)
DormNum is a foreign key (matching entity not shown)

FIGURE 6-29: Entity subtype in an E-R diagram

Nulls and Entity Subtypes (continued)

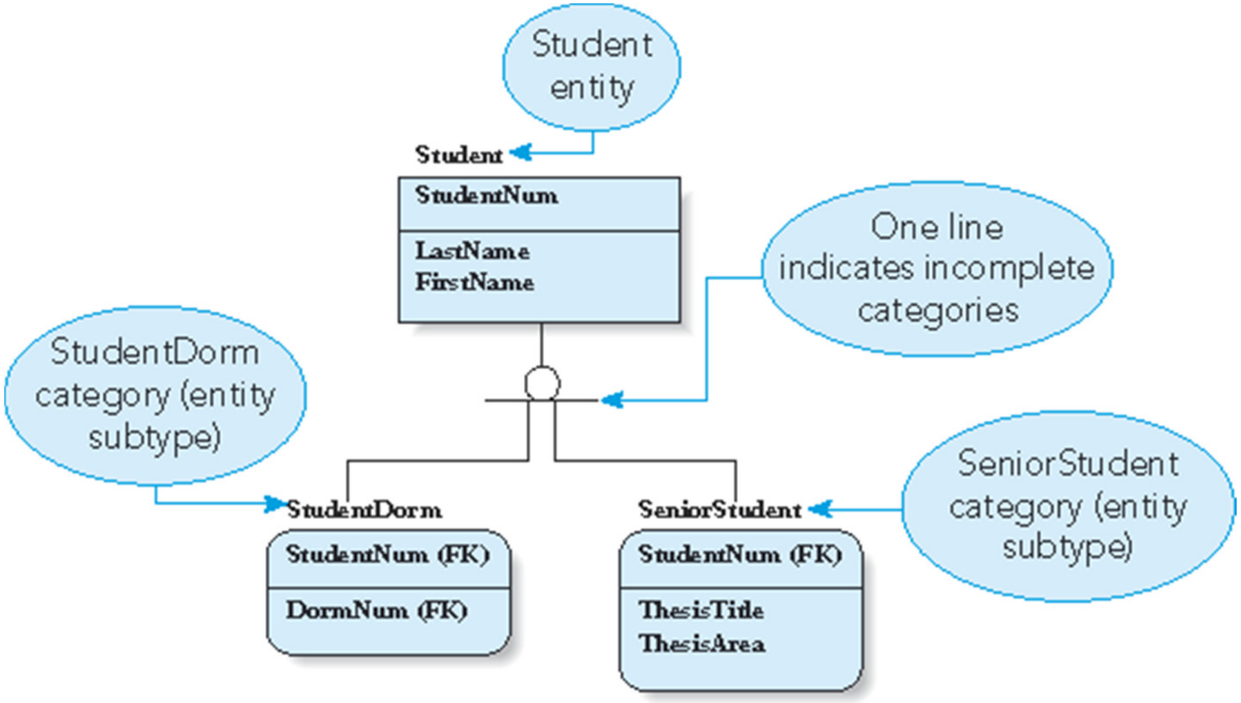


FIGURE 6-32: Two entity subtypes—incomplete categories

Nulls and Entity Subtypes (continued)

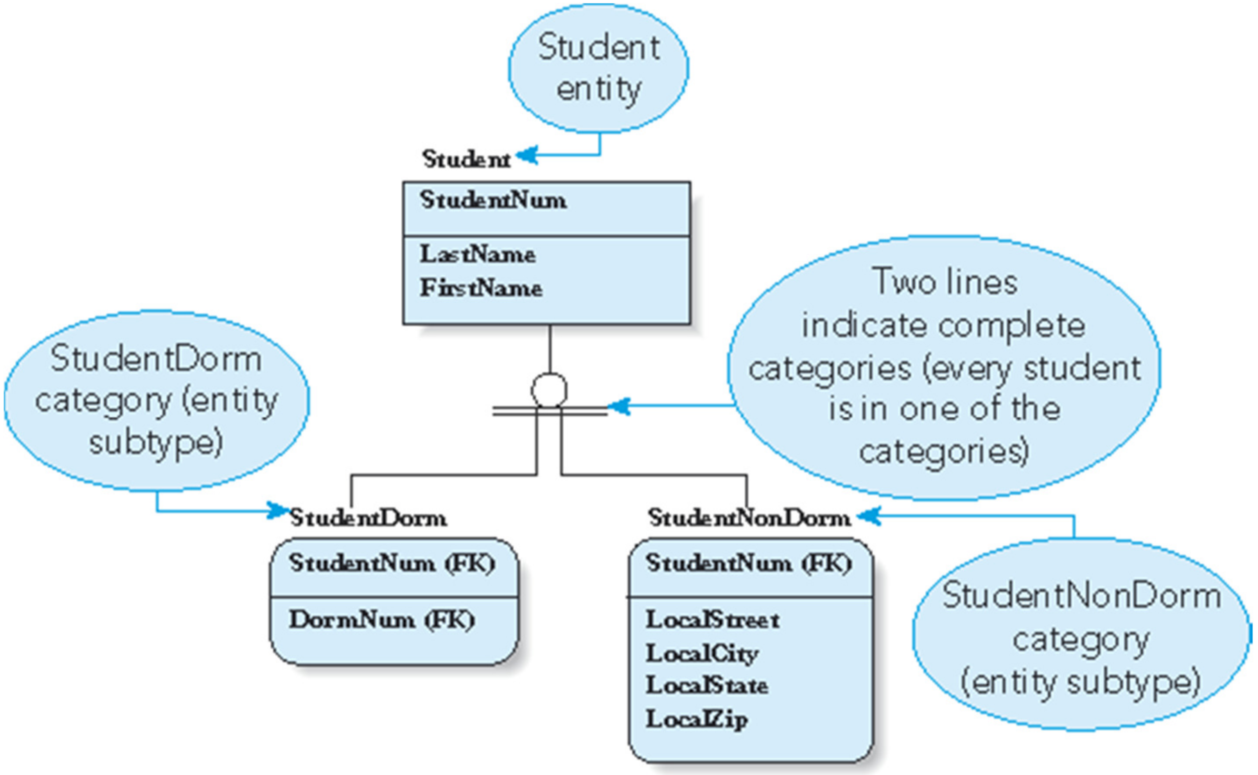

Student entity
Student
StudentNum
LastName
FirstName
StudentDorm category (entity subtype)
Two lines indicate complete categories (every student is in one of the categories)
StudentDorm
StudentNum (FK)
DormNum (FK)
StudentNonDorm
StudentNum (FK)
LocalStreet
LocalCity
LocalState
LocalZip
StudentNonDorm category (entity subtype)

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)

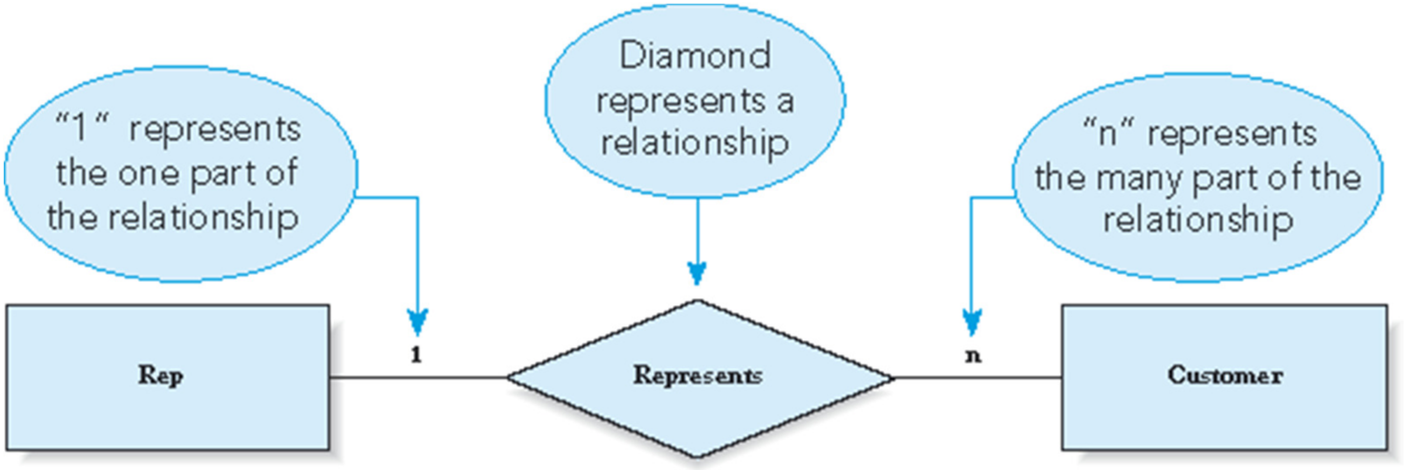

"1" represents the one part of the relationship
Diamond represents a relationship
"n" represents the many part of the relationship
Rep
1
Represents
n
Customer

FIGURE 6-34: One-to-many relationship

The Entity-Relationship Model (continued)

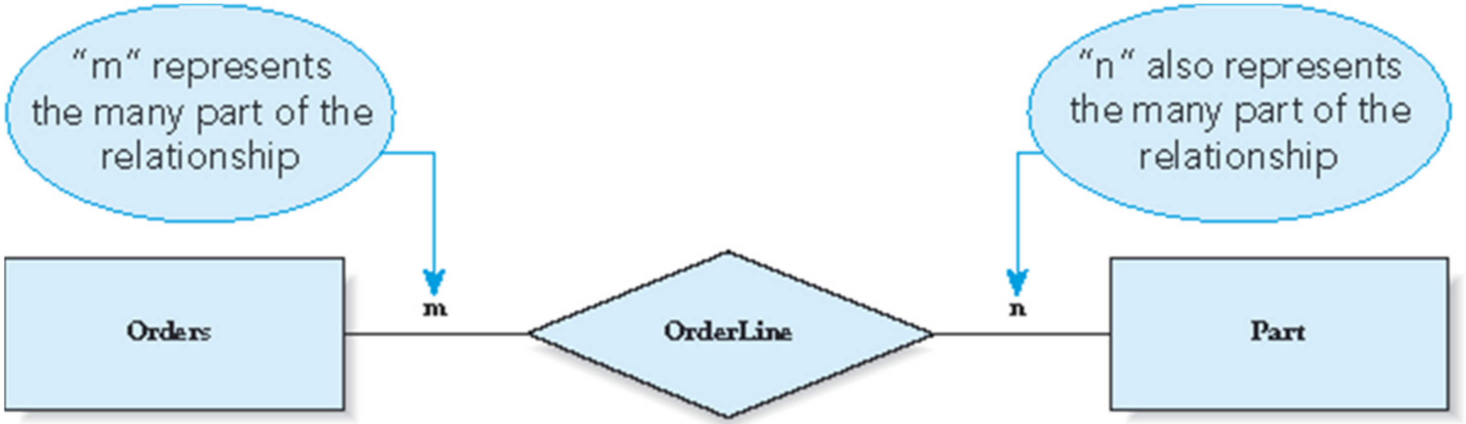

"m" represents the many part of the relationship
"n" also represents the many part of the relationship
m
n
Orders
OrderLine
Part

FIGURE 6-35: Many-to-many relationship

The Entity-Relationship Model (continued)

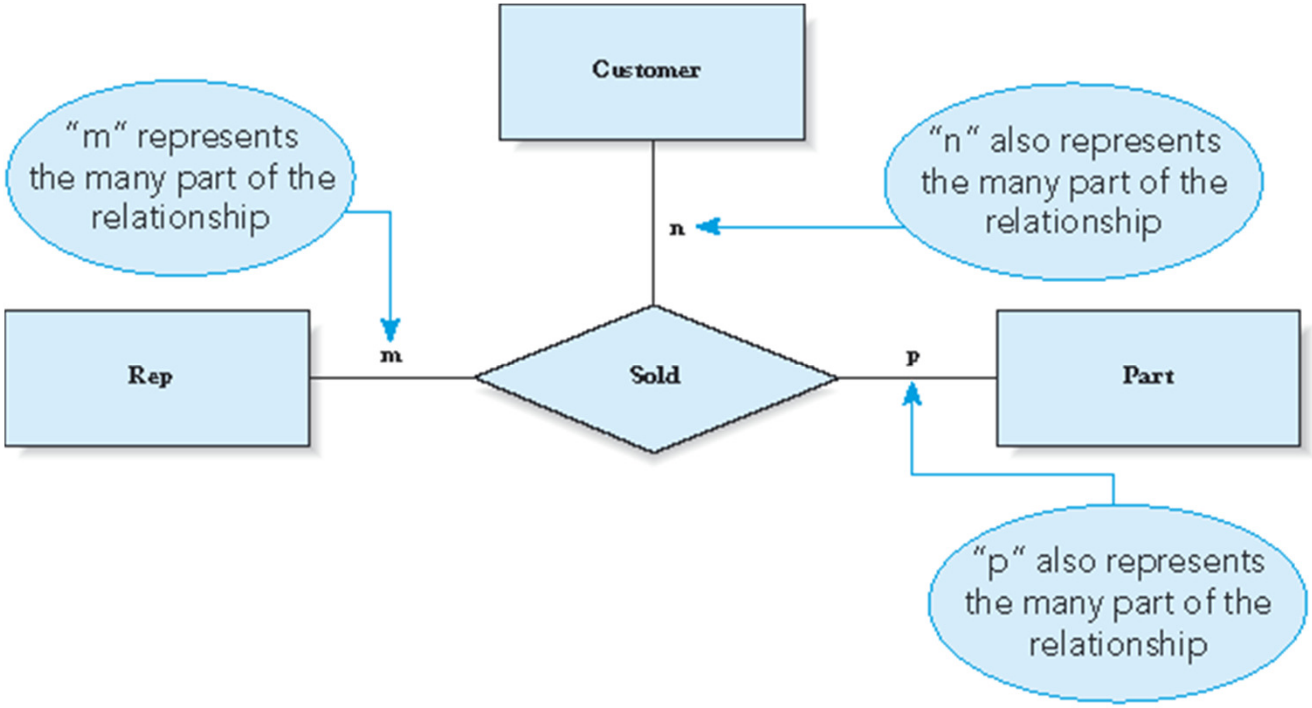

Customer
"m" represents the many part of the relationship
"n" also represents the many part of the relationship
n
m
Rep
Sold
p
Part
"p" also represents the many part of the relationship

FIGURE 6-36: Many-to-many-to-many relationship

The Entity-Relationship Model (continued)

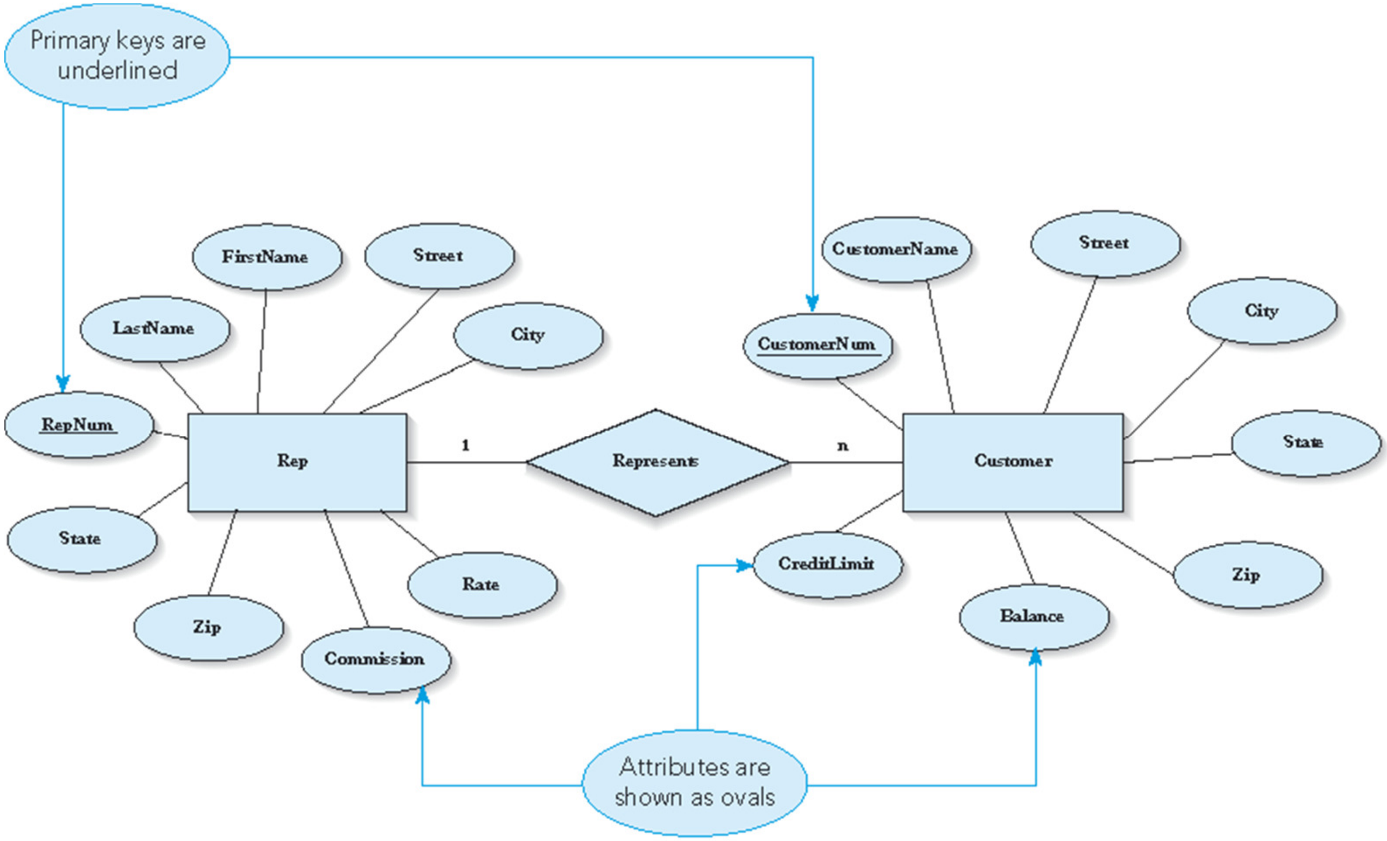


FIGURE 6-37: One-to-many relationship with attributes added

The Entity-Relationship Model (continued)

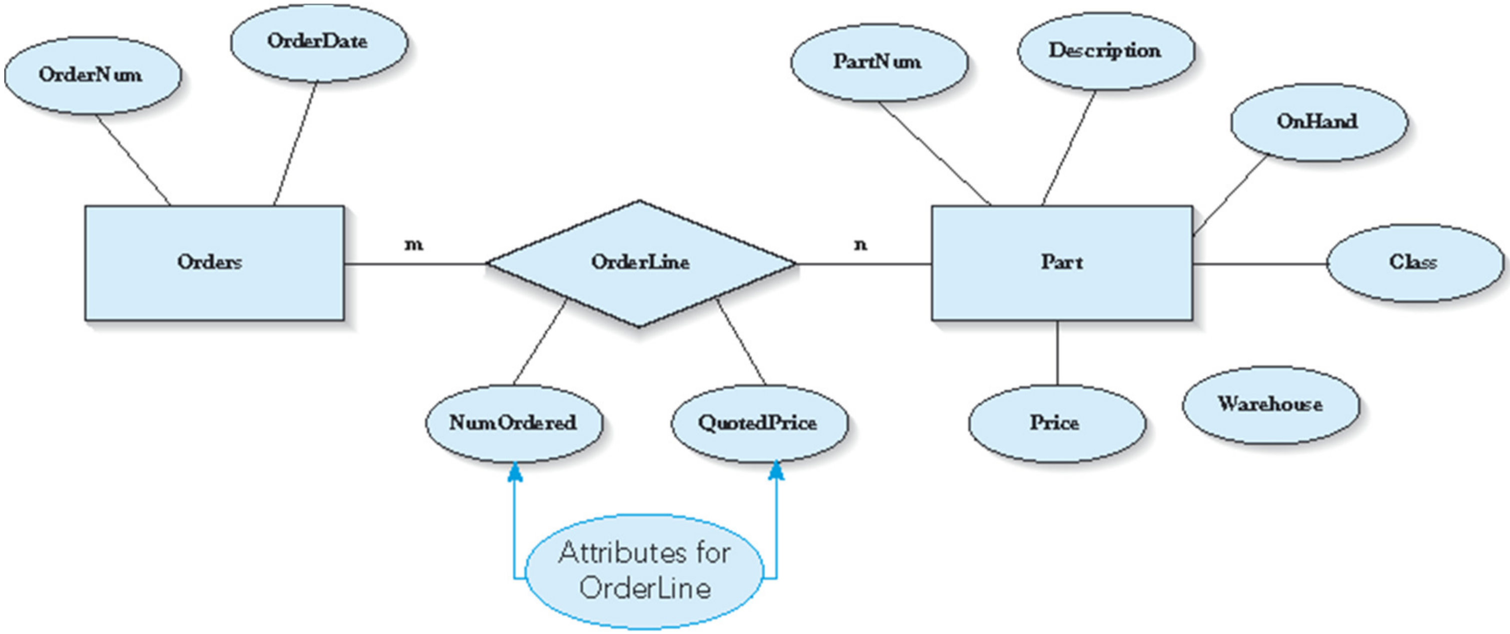


FIGURE 6-38: Many-to-many relationship with attributes

The Entity-Relationship Model (continued)

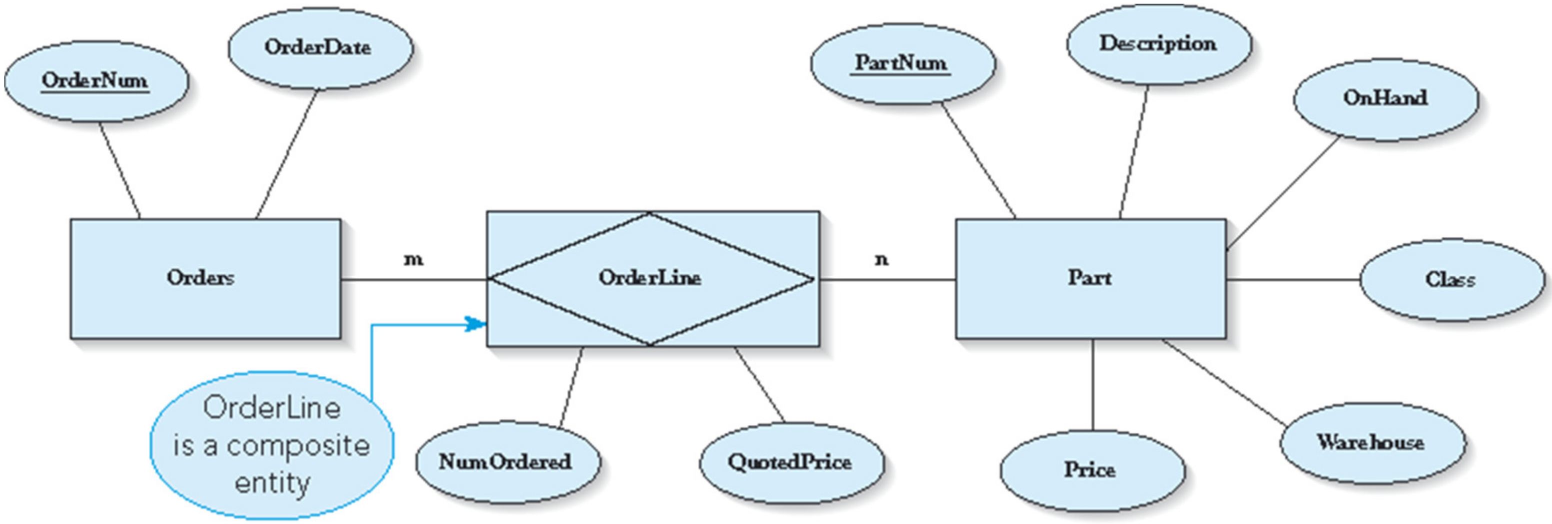


FIGURE 6-39: Composite entity

The Entity-Relationship Model (continued)

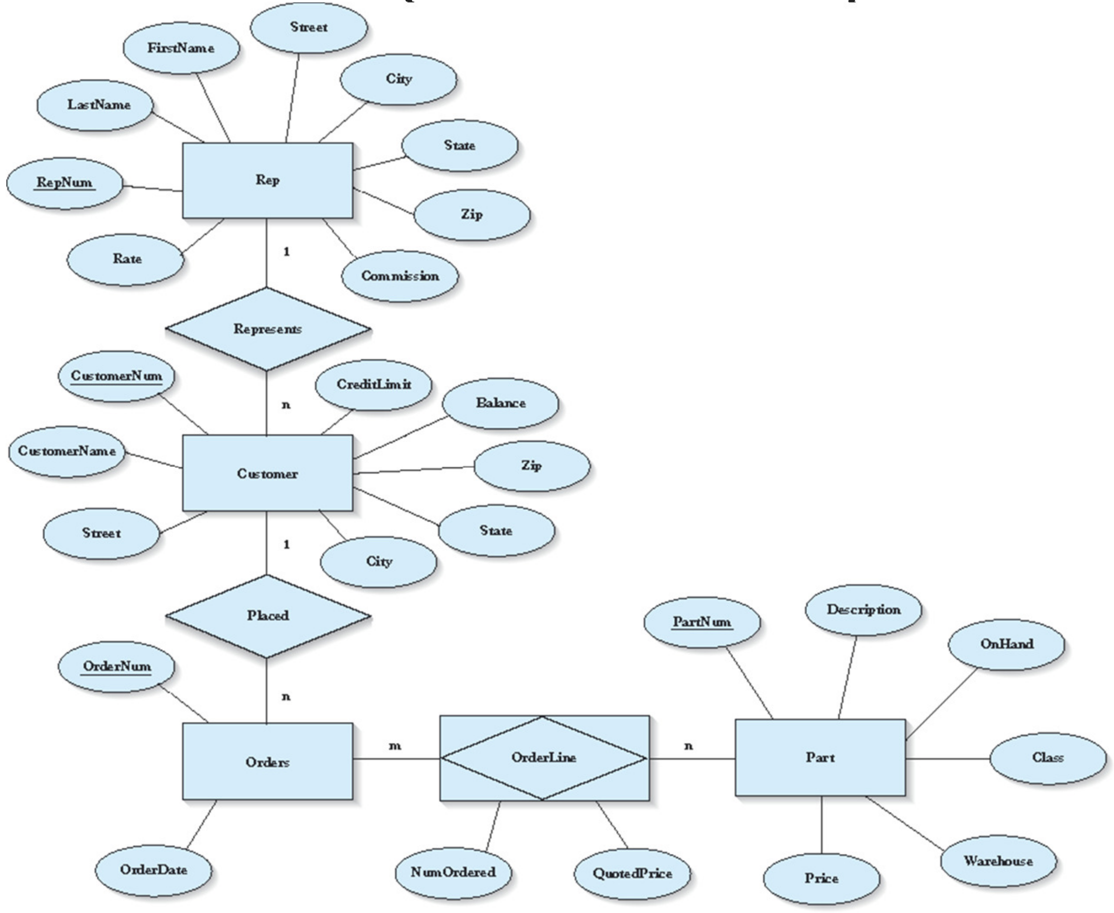


FIGURE 6-40: Complete E-R diagram for the Premiere Products database

The Entity-Relationship Model (continued)

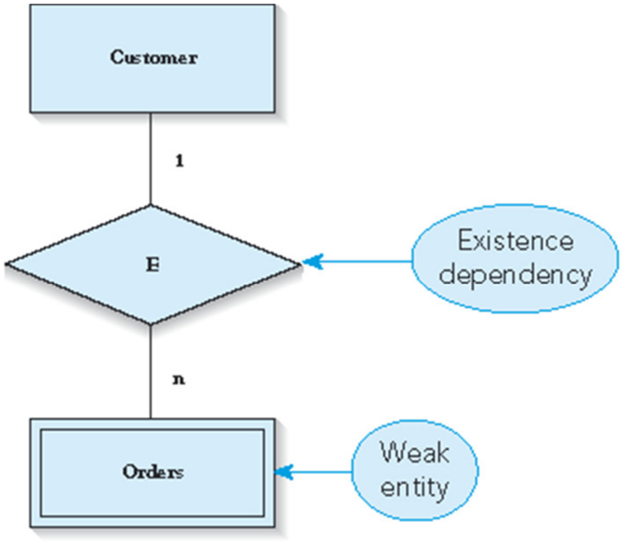


FIGURE 6-41: E-R diagram with an existence dependency and a weak entity

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)

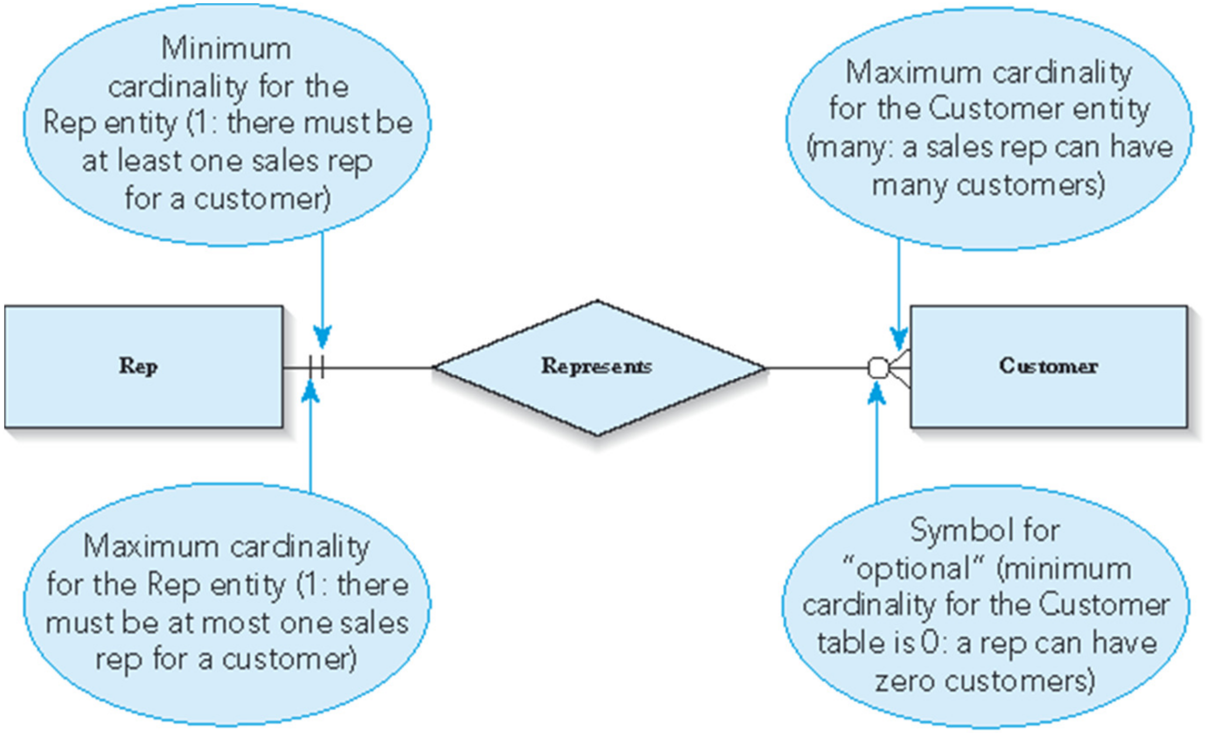

Minimum cardinality for the Rep entity (1: there must be at least one sales rep for a customer)
Maximum cardinality for the Customer entity (many: a sales rep can have many customers)
Rep
Represents
Customer
Maximum cardinality for the Rep entity (1: there must be at most one sales rep for a customer)
Symbol for "optional" (minimum cardinality for the Customer table is 0: a rep can have zero customers)

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