# 10

## Database Design 1

# 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
- 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
- Review existing documents to obtain information

# 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 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
    - include primary key of the “one” table as a foreign key in the “many” table
  - Many-to-many
    - create a new table whose primary key is the combination of the primary keys of the original tables
  - One-to-one
    - 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

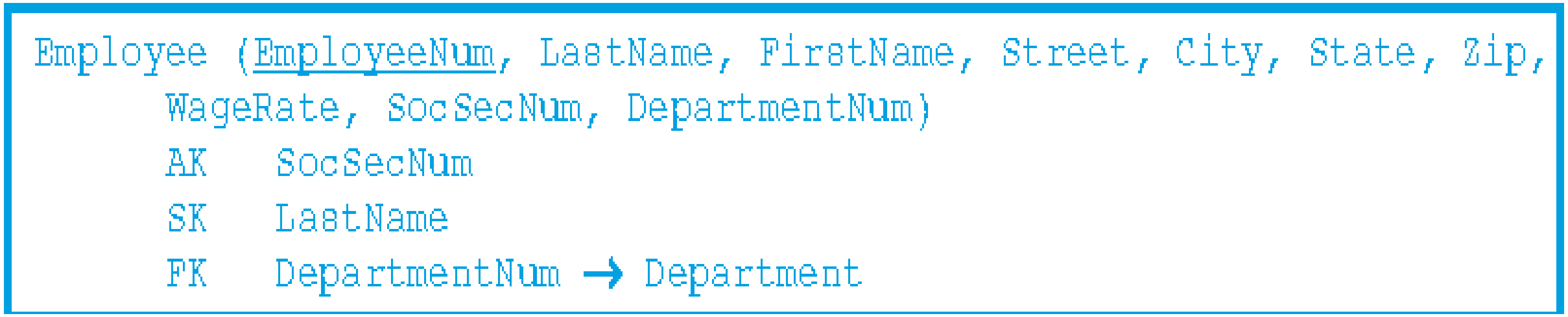
- For each table, identify:
  - Primary key
  - Alternate keys
  - Secondary keys
  - Foreign keys
- Alternate key: column(s) that could have been chosen as a primary key but was not
- **Secondary keys**: columns of interest strictly for retrieval purposes
- Foreign key: column(s) in one table that is 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



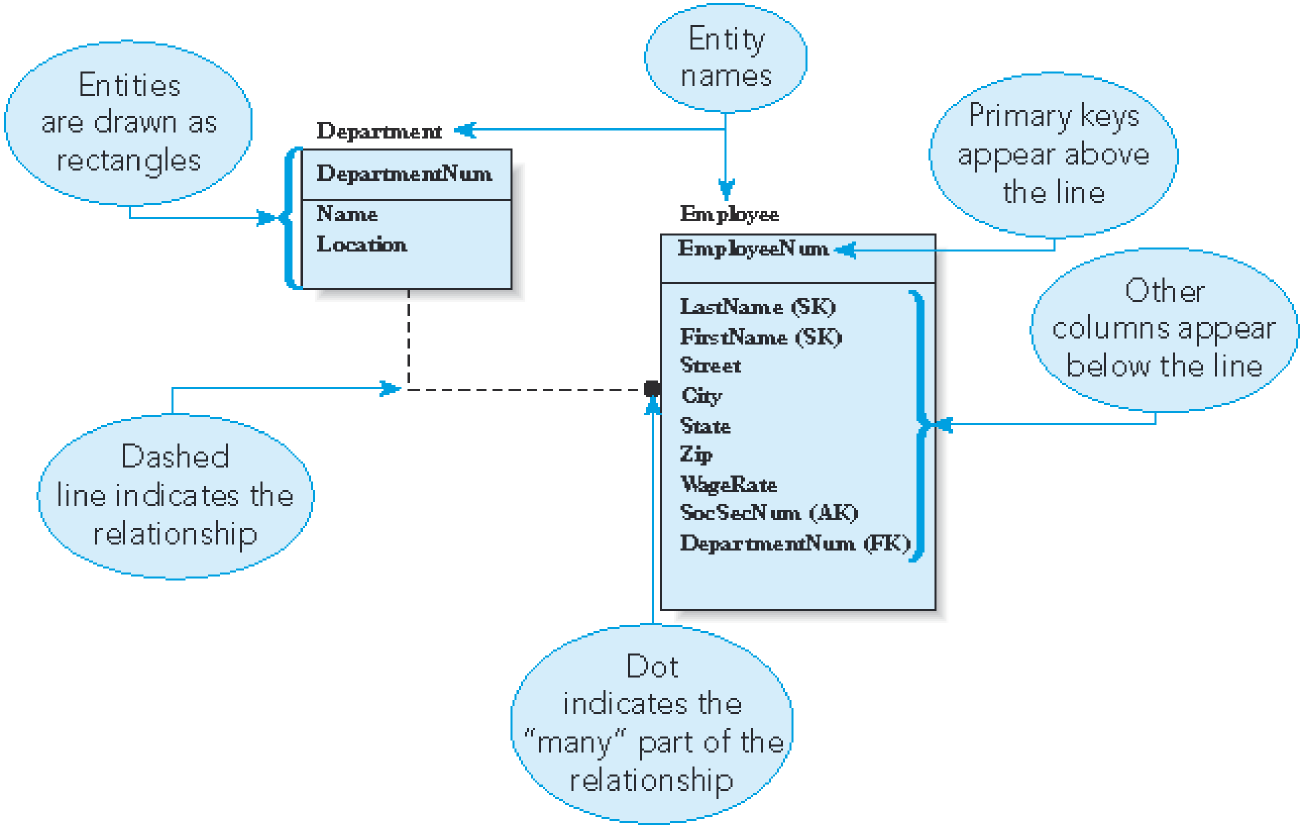
```
Employee (EmployeeNum, LastName, FirstName, Street, City, State, Zip,
      WageRate, SocSecNum, DepartmentNum)
      AK    SocSecNum
      SK    LastName
      FK    DepartmentNum → Department
```

Source: Concepts of Database Management

# 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
- 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



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)

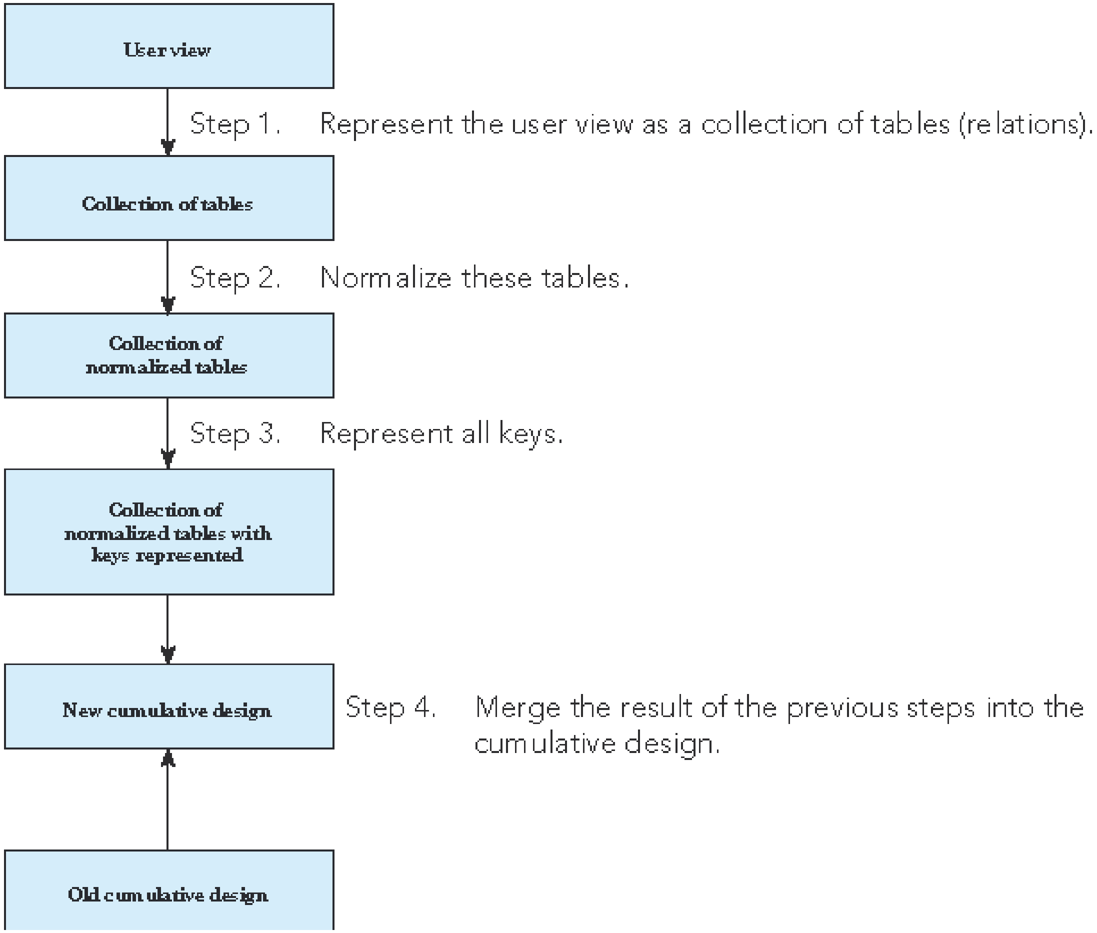
**E-R diagram**

Source: Concepts of Database Management

# 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)

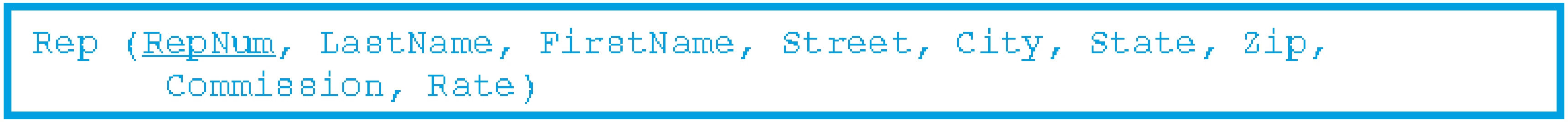

User view
Step 1. Represent the user view as a collection of tables (relations).
Collection of tables
Step 2. Normalize these tables.
Collection of normalized tables
Step 3. Represent all keys.
Collection of normalized tables with keys represented
New cumulative design
Step 4. Merge the result of the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

**Information-level design method**

Source: Concepts of Database Management

# Database Design Examples

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

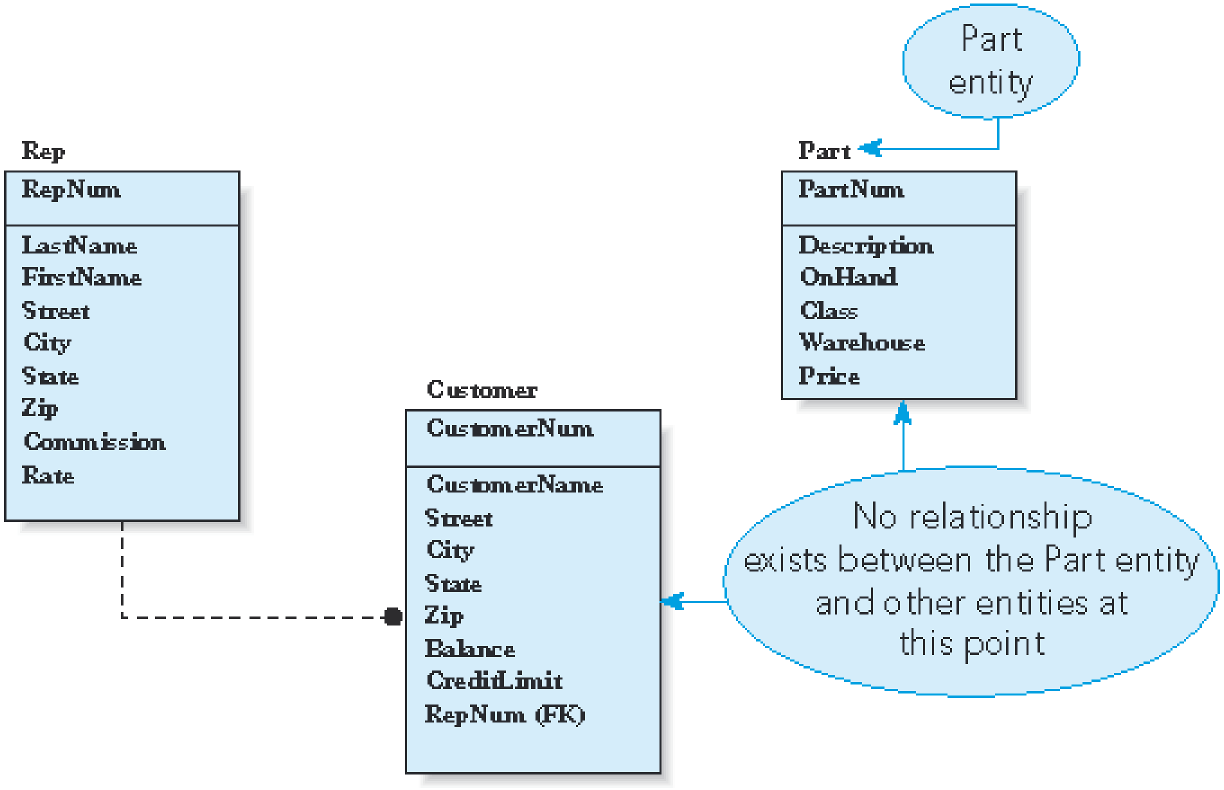


```
Rep (RepNum, LastName, FirstName, Street, City, State, Zip,
        Commission, Rate)
```

**Cumulative design after first user view**

Source: Concepts of Database Management

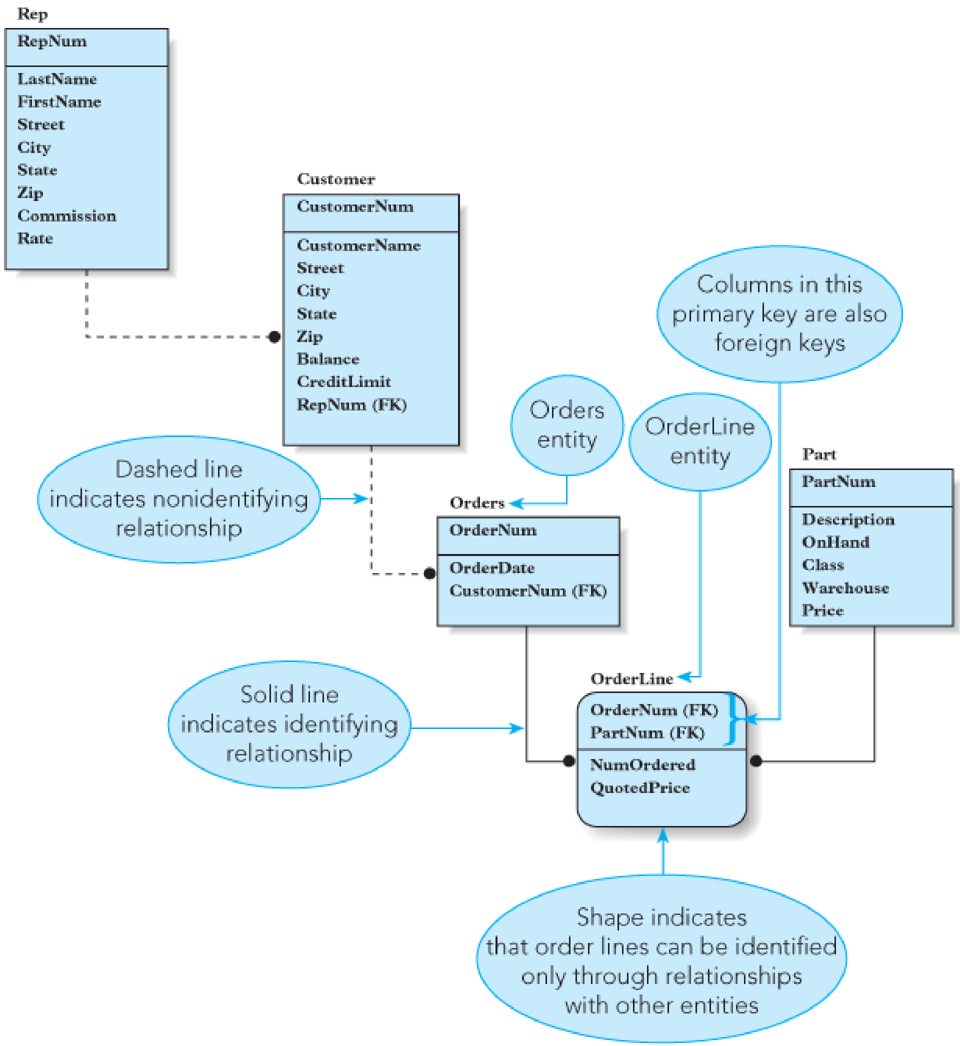
# Database Design Examples (continued)



**Cumulative design after third user view**

Source: Concepts of Database Management

# Database Design Examples (continued)



Rep
RepNum
LastName
FirstName
Street
City
State
Zip
Commission
Rate

Customer
CustomerNum
CustomerName
Street
City
State
Zip
Balance
CreditLimit
RepNum (FK)

Orders
OrderNum
OrderDate
CustomerNum (FK)

OrderLine
OrderNum (FK)
PartNum (FK)
NumOrdered
QuotedPrice

Part
PartNum
Description
OnHand
Class
Warehouse
Price

Dashed line indicates nonidentifying relationship

Solid line indicates identifying relationship

Orders entity

OrderLine entity

Columns in this primary key are also foreign keys

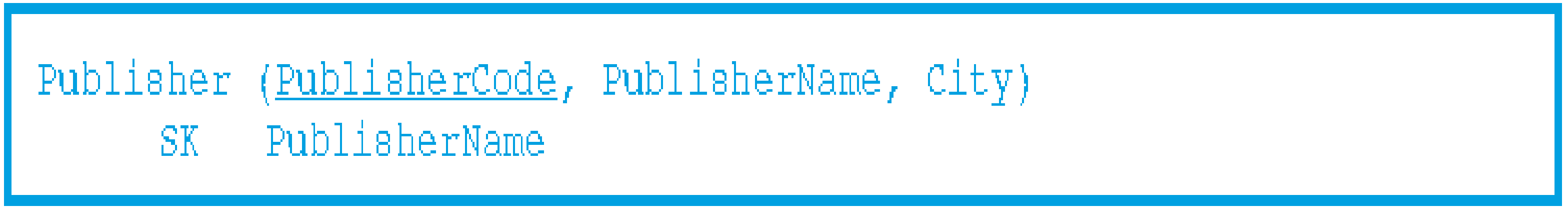
Shape indicates that order lines can be identified only through relationships with other entities

**Final information-level design**

Source: Concepts of Database Management

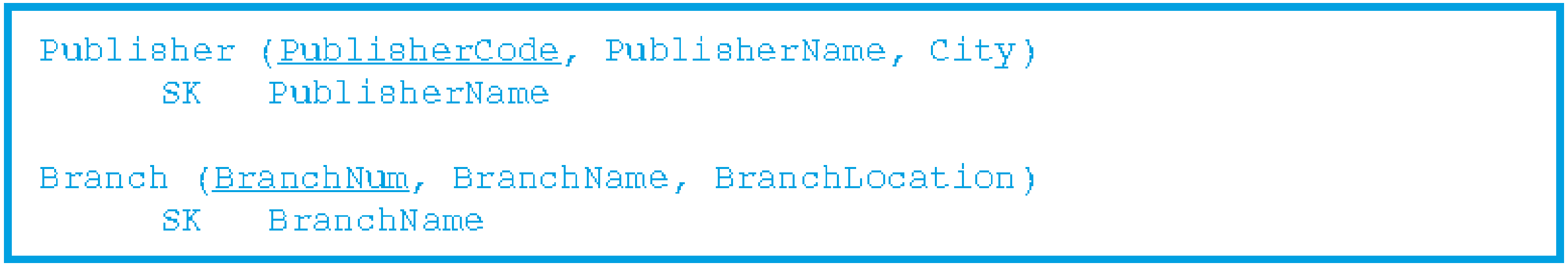
# Database Design Examples (continued)

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



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

**DBDL for Book database after first user view**

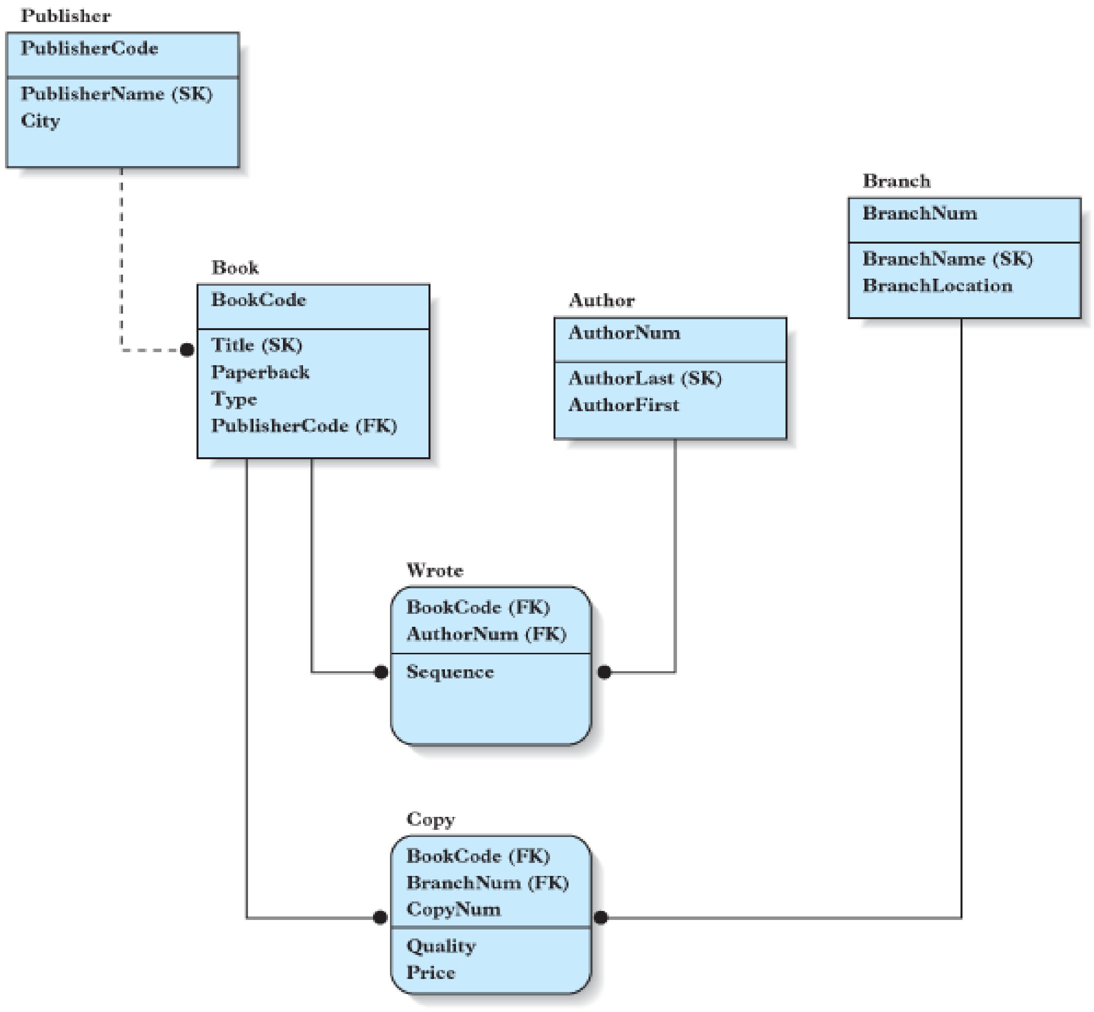


```
Publisher (PublisherCode, PublisherName, City)
      SK   PublisherName

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

**DBDL for Book database after second user view**

Source: Concepts of Database Management

# Database Design Examples (continued)



**Cumulative design after fifth user view**

Source: Concepts of Database Management

# 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

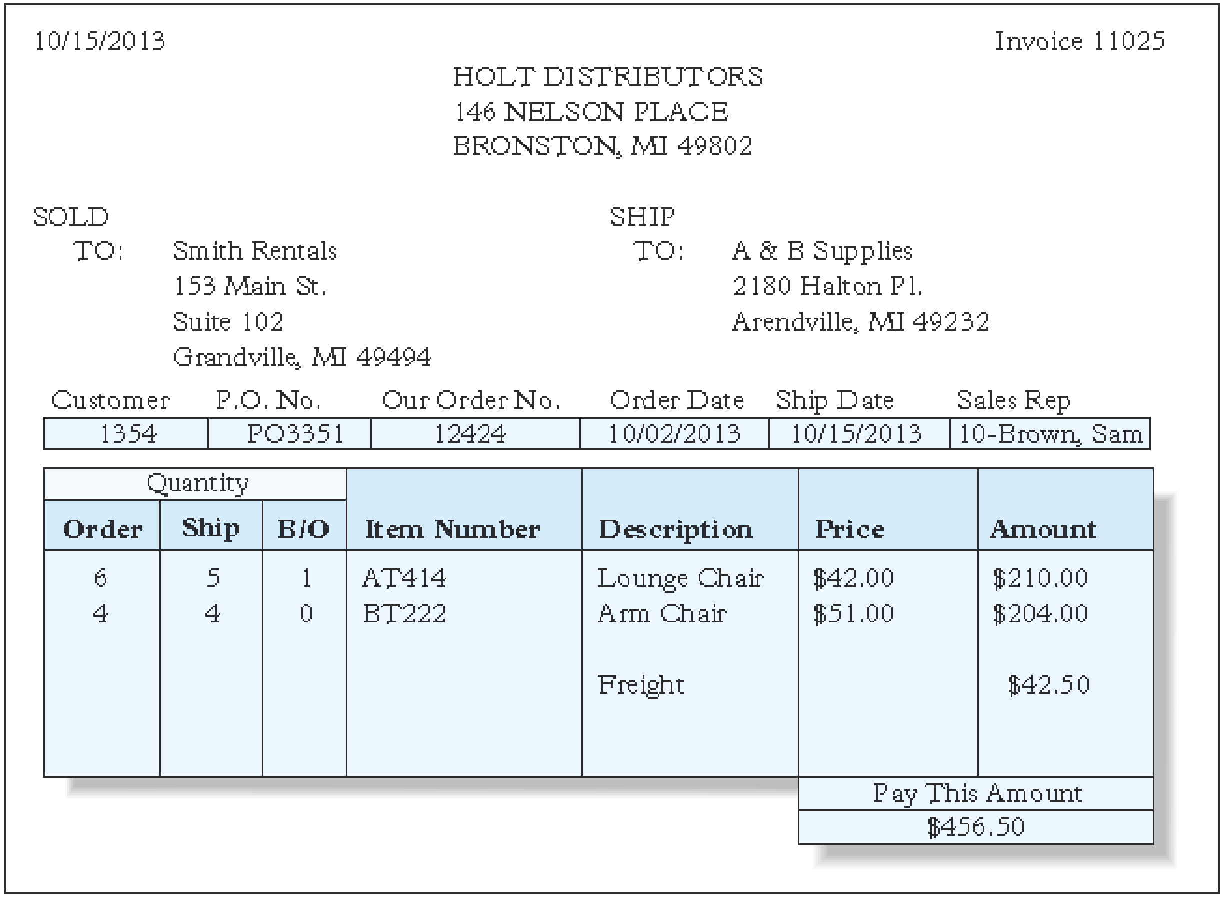
# Survey Form

- 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



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 |

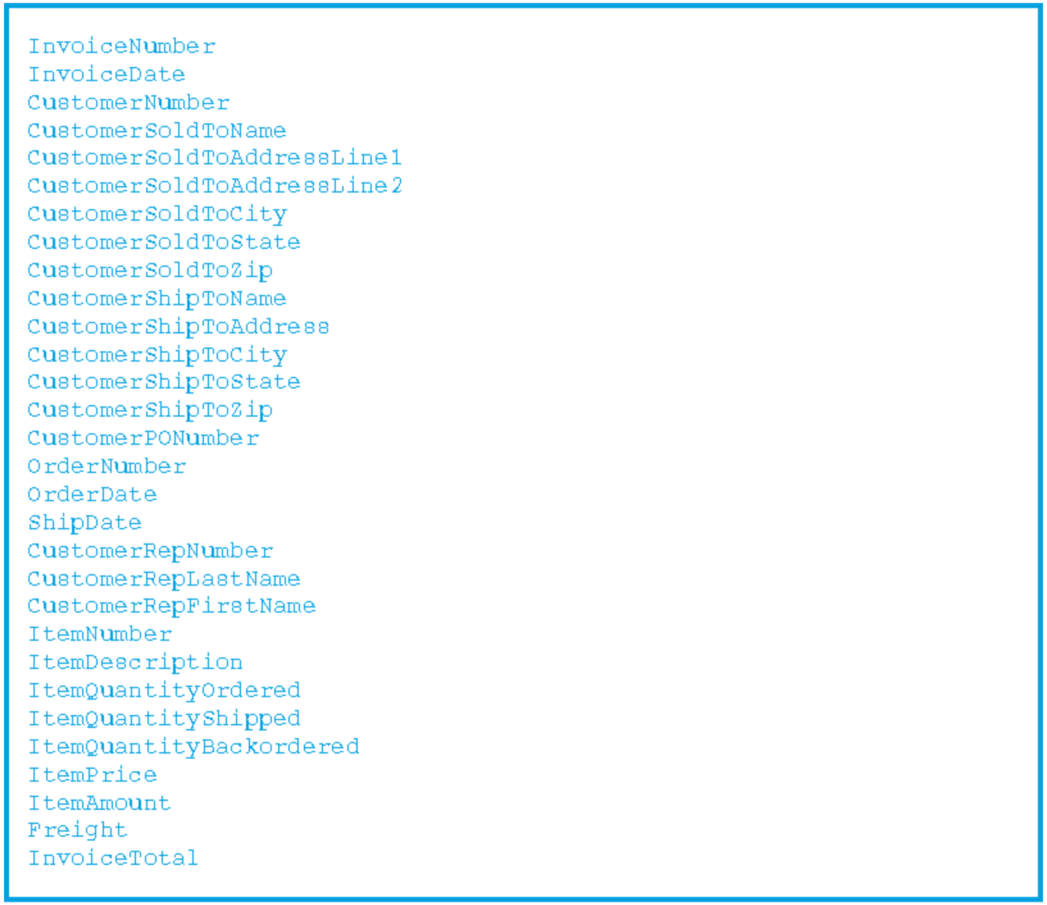
| Quantity | | | | | | |
|---|---|---|---|---|---|---|
| **Order** | **Ship** | **B/O** | **Item Number** | **Description** | **Price** | **Amount** |
| 6 | 5 | 1 | AT414 | Lounge Chair | $42.00 | $210.00 |
| 4 | 4 | 0 | BT222 | Arm Chair | $51.00 | $204.00 |
| | | | | Freight | | $42.50 |

Pay This Amount
$456.50

**Invoice for Holt Distributors**

Source: Concepts of Database Management

# Obtaining Information from Existing Documents

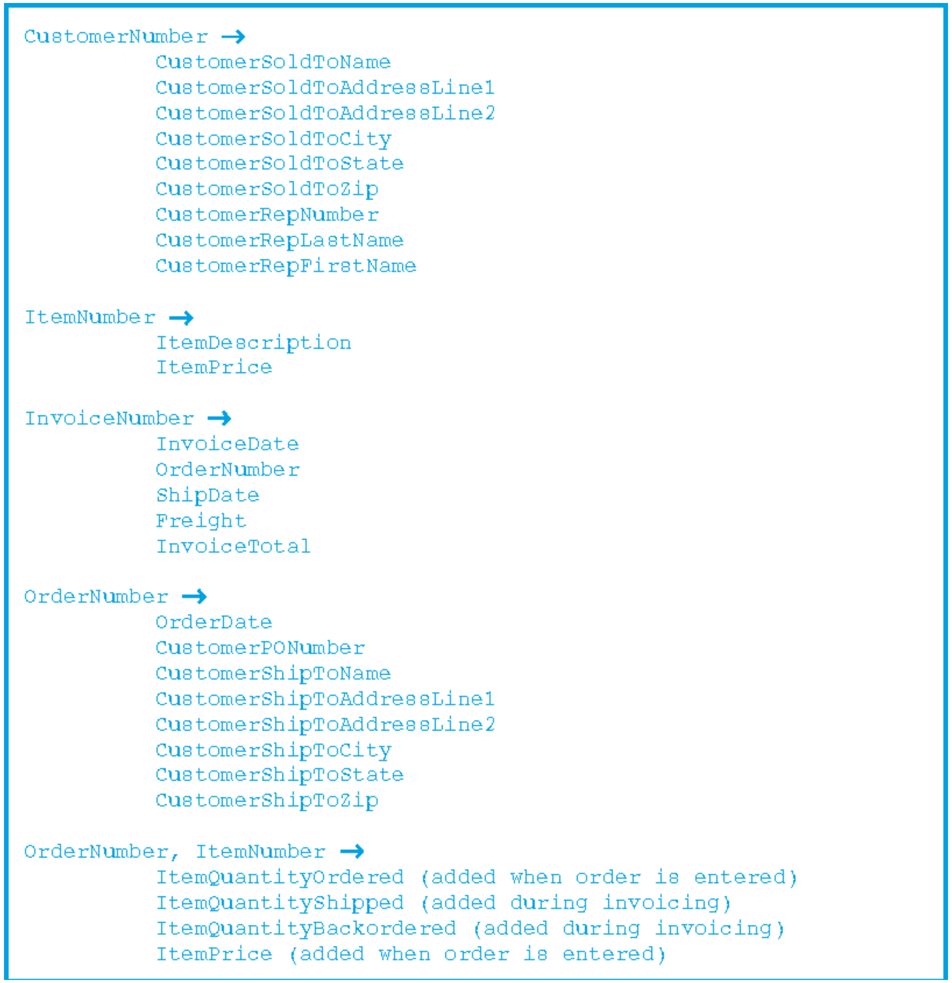


```
InvoiceNumber
InvoiceDate
CustomerNumber
CustomerSoldToName
CustomerSoldToAddressLine1
CustomerSoldToAddressLine2
CustomerSoldToCity
CustomerSoldToState
CustomerSoldToZip
CustomerShipToName
CustomerShipToAddress
CustomerShipToCity
CustomerShipToState
CustomerShipToZip
CustomerPONumber
OrderNumber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

**List of possible attributes for the Holt Distributors invoice**

Source: Concepts of Database Management

# Obtaining Information from Existing Documents

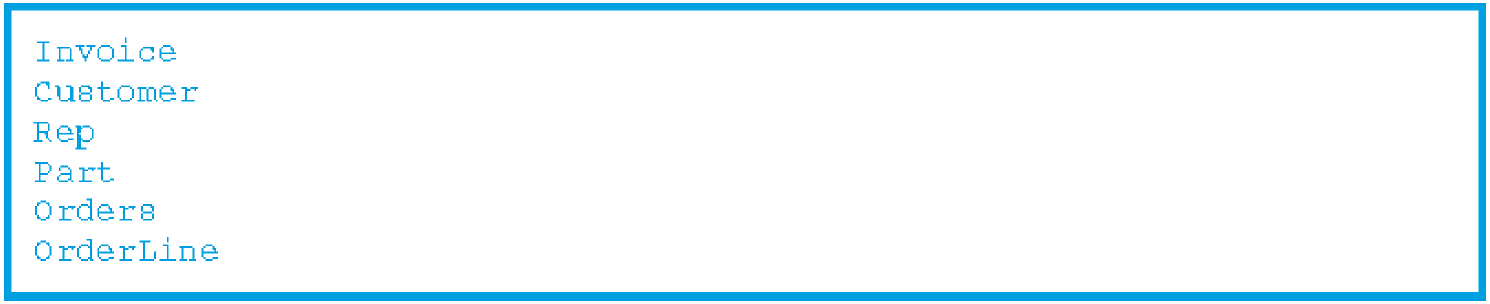


```
CustomerNumber →
          CustomerSoldToName
          CustomerSoldToAddressLine1
          CustomerSoldToAddressLine2
          CustomerSoldToCity
          CustomerSoldToState
          CustomerSoldToZip
          CustomerRepNumber
          CustomerRepLastName
          CustomerRepFirstName

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)
```

**Revised list of functional dependencies for the Holt Distributors invoice**

Source: Concepts of Database Management

# Obtaining Information from Existing Documents



Invoice
Customer
Rep
Part
Orders
OrderLine

**Expanded list of entities**

Source: Concepts of Database Management

# Summary

- Database design is a two-part process:
  - information-level design (not dependent on a particular DBMS)
  - 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
- Database design is represented in Database Design Language (DBDL)
- Designs can be represented visually using 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
- 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