



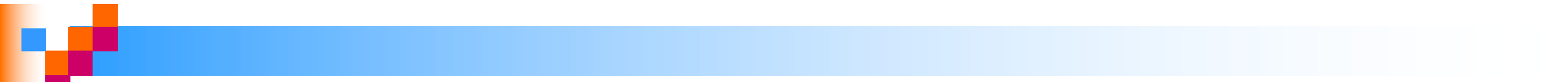
Modern Systems Analysis and Design

Seventh Edition

Jeffrey A. Hoffer
Joey F. George
Joseph S. Valacich

Chapter 9

Designing Databases



Learning Objectives

- ✓ Concisely define each of the following key database design terms: relation, primary key, normalization, functional dependency, foreign key, referential integrity, field, data type, null value, denormalization, file organization, index, and secondary key.
- ✓ Explain the role of designing databases in the analysis and design of an information system.
- ✓ Transform an entity-relationship (E-R) diagram into an equivalent set of well-structured (normalized) relations.



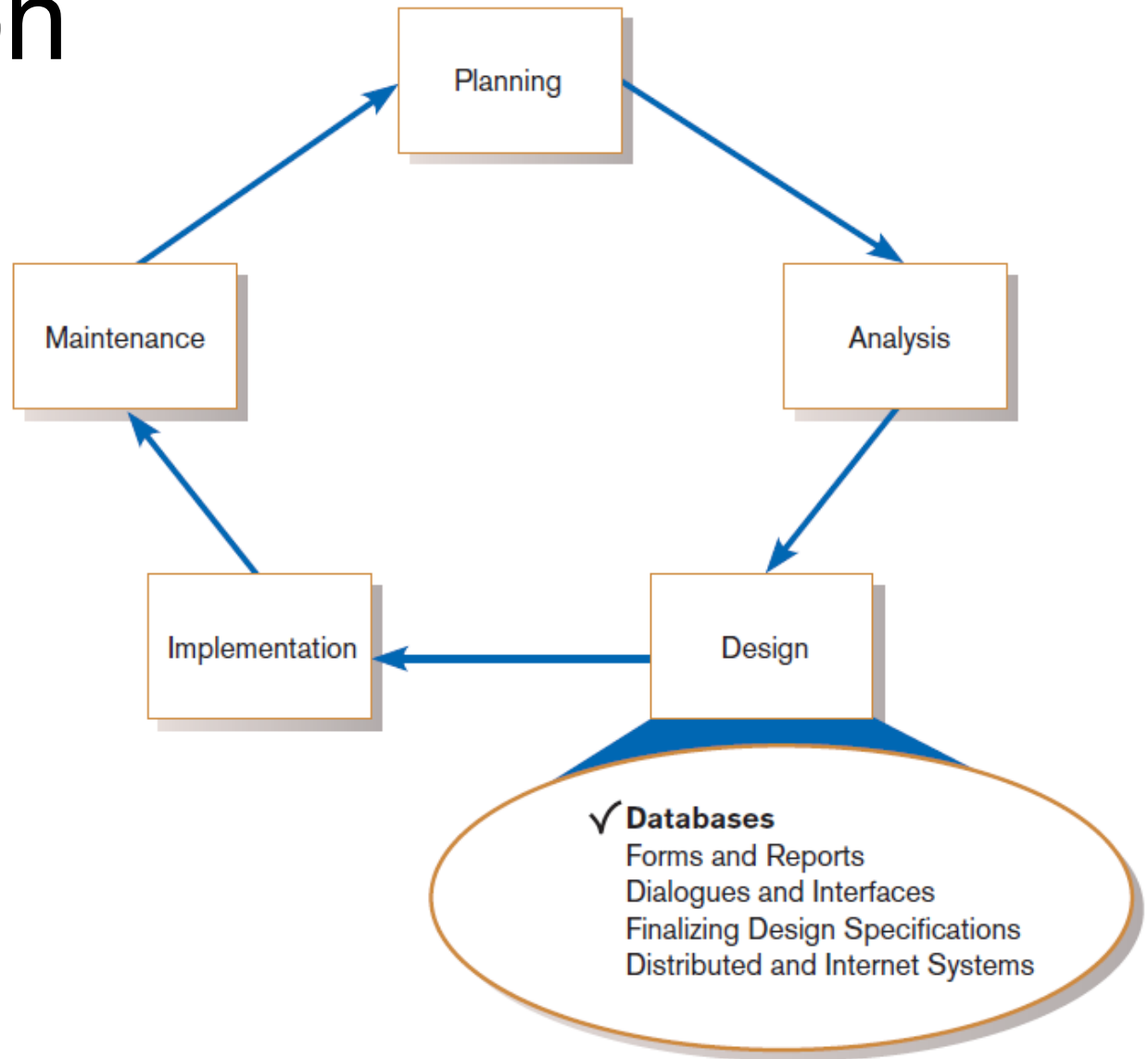
Learning Objectives (Cont.)

- ✓ Merge normalized relations from separate user views into a consolidated set of well-structured relations.
- ✓ Choose storage formats for fields in database tables.
- ✓ Translate well-structured relations into efficient database tables.
- ✓ Explain when to use different types of file organizations to store computer files.
- ✓ Describe the purpose of indexes and the important considerations in selecting attributes to be indexed.



Introduction

FIGURE 9-1
Systems development
life cycle with design
phase highlighted





Database Design

- File and database design occurs in two steps.
 1. Develop a logical database model, which describes data using notation that corresponds to a data organization used by a database management system.
 - Relational database model
 2. Prescribe the technical specifications for computer files and databases in which to store the data.
 - Physical database design provides specifications
- Logical and physical database design in parallel with other system design steps

The Process of Database Design

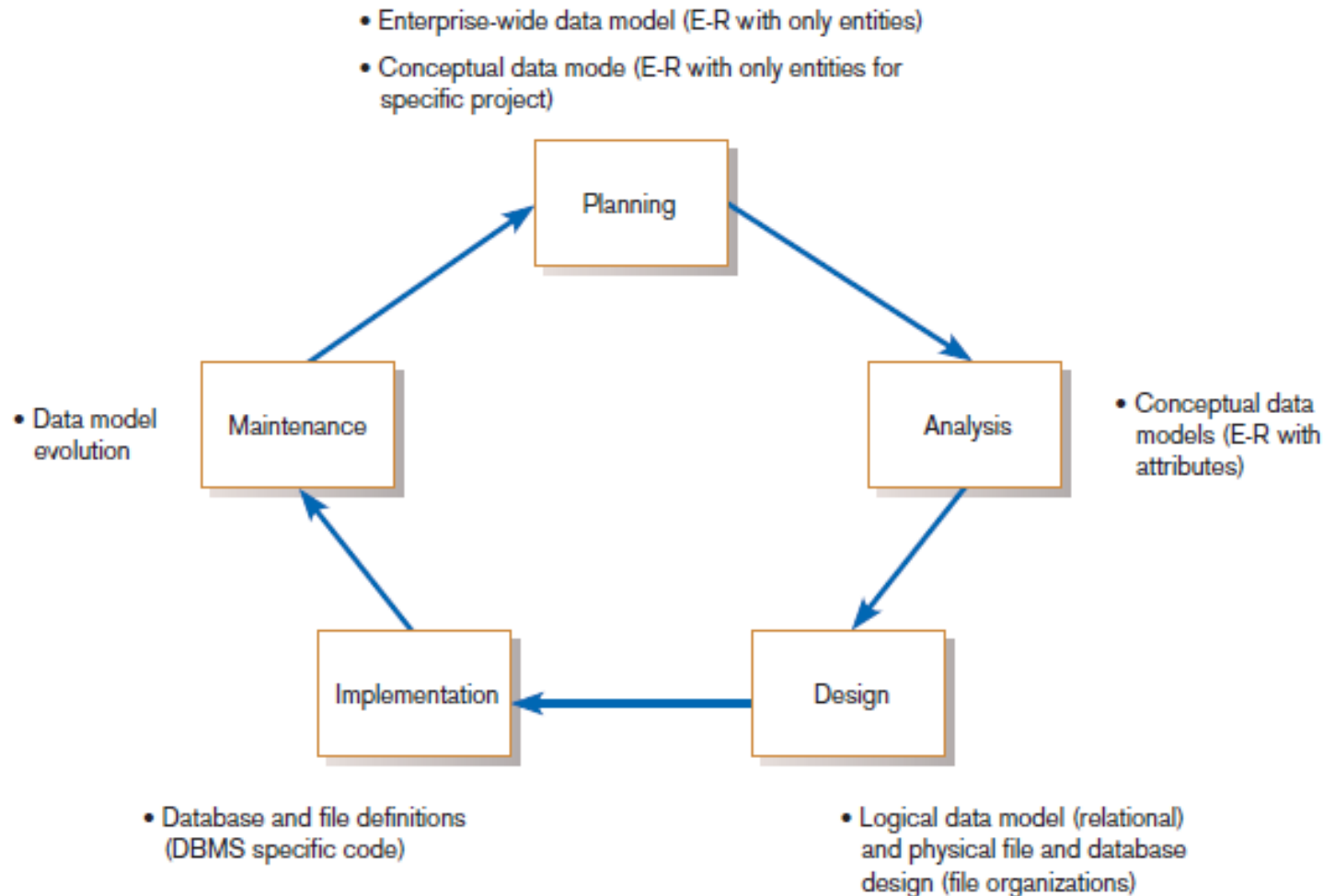
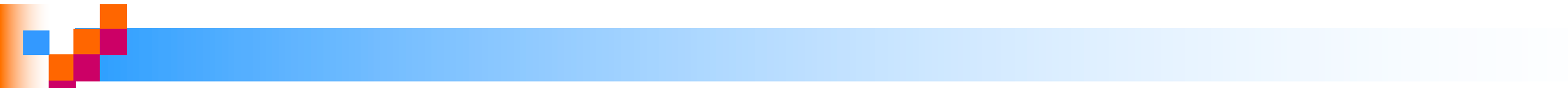


FIGURE 9-2

Relationship between data modeling and the systems development life cycle



The Process of Database Design (Cont.)

- Four key steps in logical database modeling and design:
 1. Develop a logical data model for each known user interface for the application using normalization principles.
 2. Combine normalized data requirements from all user interfaces into one consolidated logical database model (view integration).
 3. Translate the conceptual E-R data model for the application into normalized data requirements.
 4. Compare the consolidated logical database design with the translated E-R model and produce one final logical database model for the application.



Physical Database Design

- Key physical database design decisions include:
 - Choosing a storage format for each attribute from the logical database model.
 - Grouping attributes from the logical database model into physical records.
 - Arranging related records in secondary memory (hard disks and magnetic tapes) so that records can be stored, retrieved and updated rapidly.
 - Selecting media and structures for storing data to make access more efficient.



Deliverables and Outcomes

- Logical database design

- Must account for every data element on a system input or output
 - Normalized relations are the primary deliverable.

- Physical database design

- Converts relations into database tables
 - Programmers and database analysts code the definitions of the database.
 - Written in Structured Query Language (SQL)

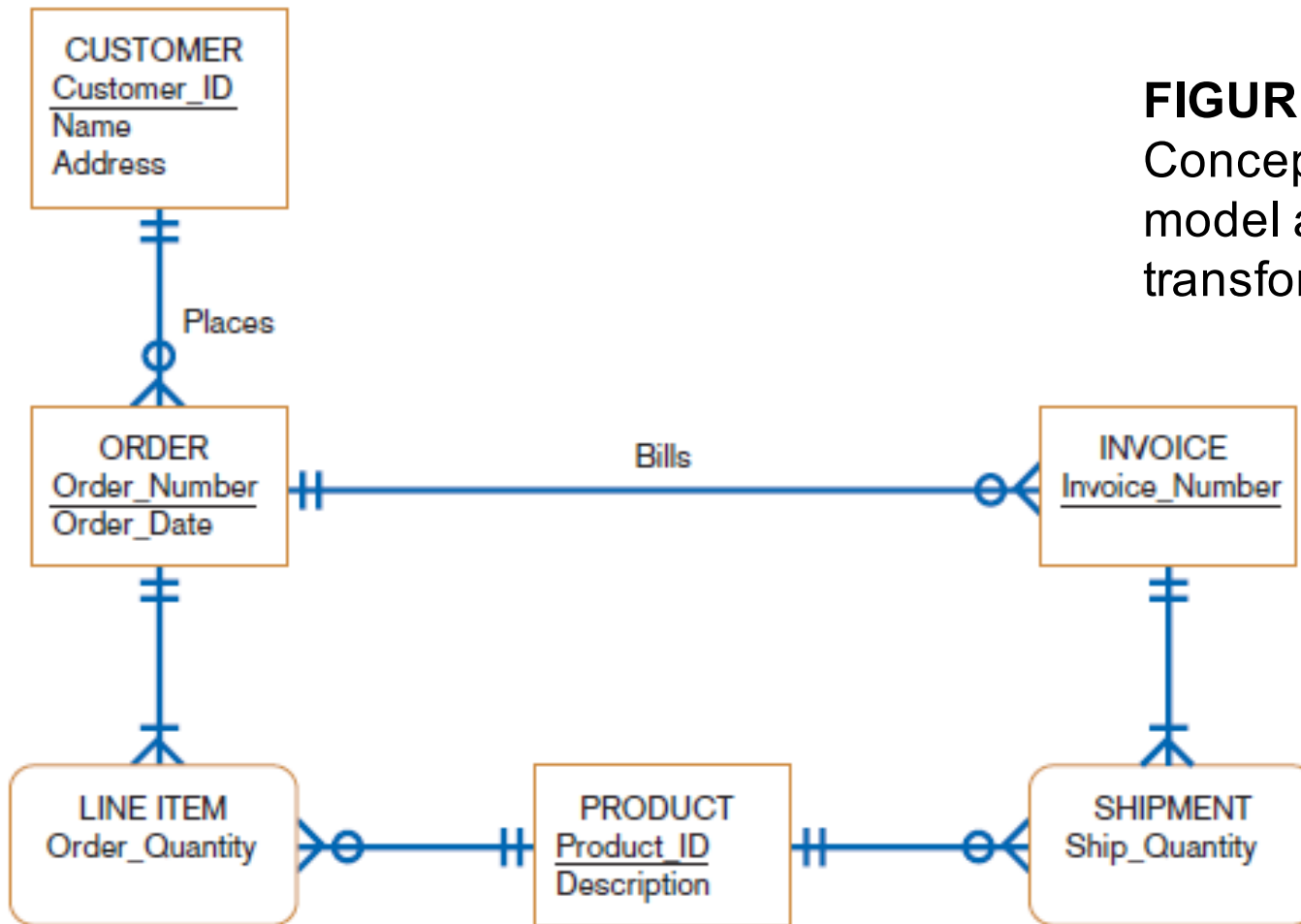


FIGURE 9-3 (d)
Conceptual data
model and
transformed relations

Relations:

```
CUSTOMER(Customer_ID,Name,Address)
PRODUCT(Product_ID,Description)
ORDER(Order_Number,Customer_ID,Order_Date)
LINE ITEM(Order_Number,Product_ID,Order_Quantity)
INVOICE(Invoice_Number,Order_Number)
SHIPMENT(Invoice_Number,Product_ID,Ship_Quantity)
```



Relational Database Model

- **Relational database model:** data represented as a set of related tables or relations
- **Relation:** a named, two-dimensional table of data; each relation consists of a set of named columns and an arbitrary number of unnamed rows



Relational Database Model (Cont.)

- Relations have several properties that distinguish them from nonrelational tables:
 - Entries in cells are simple.
 - Entries in columns are from the same set of values.
 - Each row is unique.
 - The sequence of columns can be interchanged without changing the meaning or use of the relation.
 - The rows may be interchanged or stored in any sequence.



Well-Structured Relation and Primary Keys

- **Well-Structured Relation (or table)**

- A relation that contains a minimum amount of redundancy
- Allows users to insert, modify, and delete the rows without errors or inconsistencies

- **Primary Key**

- An attribute whose value is unique across all occurrences of a relation

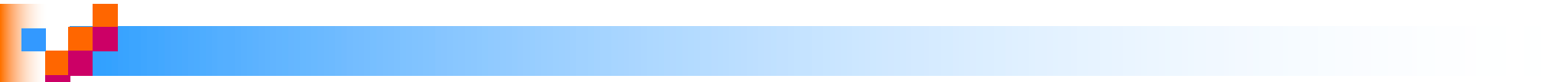
- **All relations have a primary key.**

- This is how rows are ensured to be unique.
- A primary key may involve a single attribute or be composed of multiple attributes.



Normalization and Rules of Normalization

- **Normalization:** the process of converting complex data structures into simple, stable data structures
- The result of normalization is that every nonprimary key attribute depends upon the whole primary key.



Normalization and Rules of Normalization (Cont.)

- First Normal Form (1NF)
 - Unique rows, no multivalued attributes
 - All relations are in 1NF
- Second Normal Form (2NF)
 - Each nonprimary key attribute is identified by the whole key (called full functional dependency)
- Third Normal Form (3NF)
 - Nonprimary key attributes do not depend on each other (i.e. no transitive dependencies)



Functional Dependencies and Primary Keys

- **Functional Dependency:** a particular relationship between two attributes
 - For a given relation, attribute B is functionally dependent on attribute A if, for every valid value of A, that value of A uniquely determines the value of B.
 - The functional dependence of B on A is represented by $A \rightarrow B$.



Functional Dependencies and Primary Keys (Cont.)

- Functional dependency is not a mathematical dependency.
- Instances (or sample data) in a relation do not prove the existence of a functional dependency.
- Knowledge of problem domain is most reliable method for identifying functional dependency.



Second Normal Form (2NF)

- A relation is in second normal form (2NF) if any of the following conditions apply:
 - The primary key consists of only one attribute.
 - No nonprimary key attributes exist in the relation.
 - Every nonprimary key attribute is functionally dependent on the full set of primary key attributes.
- To convert a relation into 2NF, decompose the relation into new relations using the attributes, called *determinants*, that determine other attributes.
- The determinants are the primary keys of the new relations.



Third Normal Form (3NF)

- A relation is in third normal form (3NF) if it is in second normal form (2NF) and there are no functional (transitive) dependencies between two (or more) nonprimary key attributes.

SALES

| <u>Customer_ID</u> | Customer_Name | Salesperson | Region |
|--------------------|---------------|-------------|--------|
| 8023 | Anderson | Smith | South |
| 9167 | Bancroft | Hicks | West |
| 7924 | Hobbs | Smith | South |
| 6837 | Tucker | Hernandez | East |
| 8596 | Eckersley | Hicks | West |
| 7018 | Arnold | Faulb | North |

FIGURE 9-9

Removing transitive dependencies
 (a) Relation with transitive dependency

SALES1

| <u>Customer_ID</u> | Customer_Name | <u>Salesperson</u> |
|--------------------|---------------|--------------------|
| 8023 | Anderson | Smith |
| 9167 | Bancroft | Hicks |
| 7924 | Hobbs | Smith |
| 6837 | Tucker | Hernandez |
| 8596 | Eckersley | Hicks |
| 7018 | Arnold | Faulb |

SPERSON

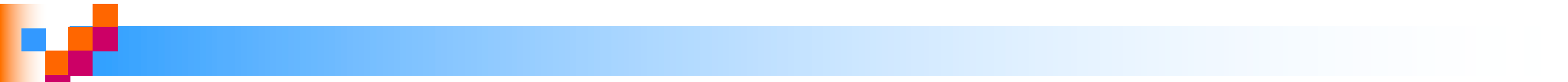
(b) Relation in 3NF

| <u>Salesperson</u> | Region |
|--------------------|--------|
| Smith | South |
| Hicks | West |
| Hernandez | East |
| Faulb | North |



Third Normal Form (3NF) (Cont.)

- **Foreign Key:** an attribute that appears as a nonprimary key attribute in one relation and as a primary key attribute (or part of a primary key) in another relation
- **Referential Integrity:** an integrity constraint specifying that the value (or existence) of an attribute in one relation depends on the value (or existence) of the same attribute in another relation



Transforming E-R Diagrams into Relations

- It is useful to transform the conceptual data model into a set of normalized relations.
- Steps
 - *Represent entities.*
 - *Represent relationships.*
 - *Normalize the relations.*
 - *Merge the relations.*



Representing Entities

- Each regular entity is transformed into a relation.
- The identifier of the entity type becomes the primary key of the corresponding relation.



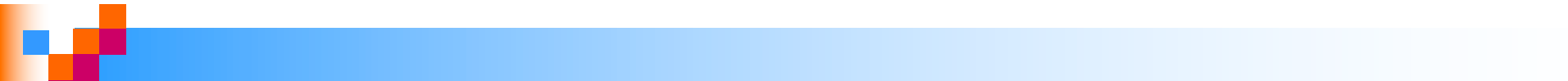
Representing Entities

- The primary key must satisfy the following two conditions.
 - The value of the key must uniquely identify every row in the relation.
 - The key should be nonredundant.
- The entity type label is translated into a relation name.



Binary 1:N and 1:1 Relationships

- The procedure for representing relationships depends on both the degree of the relationship—unary, binary, ternary—and the cardinalities of the relationship.
- **Binary 1:N Relationship** is represented by adding the primary key attribute (or attributes) of the entity on the one side of the relationship as a foreign key in the relation that is on the many side of the relationship.



Binary 1:N and 1:1 Relationships (Cont.)

- **Binary or Unary 1:1 Relationship is represented by any of the following choices:**
 - Add the primary key of A as a foreign key of B.
 - Add the primary key of B as a foreign key of A.
 - Both of the above

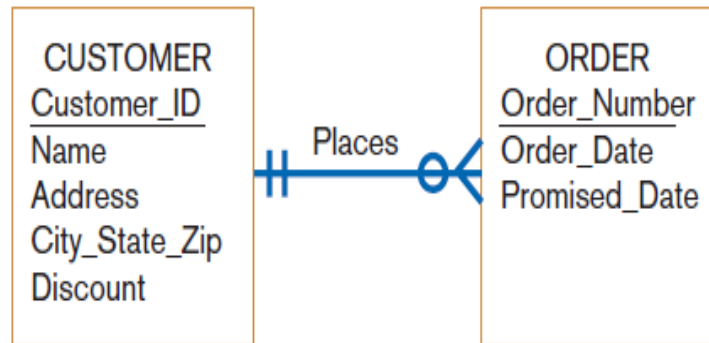


FIGURE 9-11
Representing a 1:N relationship
(a) E-R diagram

CUSTOMER

| <u>Customer_ID</u> | Name | Address | City_State_ZIP | Discount |
|--------------------|----------------------|----------------|-----------------------|----------|
| 1273 | Contemporary Designs | 123 Oak St. | Austin, TX 28384 | 5% |
| 6390 | Casual Corner | 18 Hoosier Dr. | Bloomington, IN 45821 | 3% |

ORDER

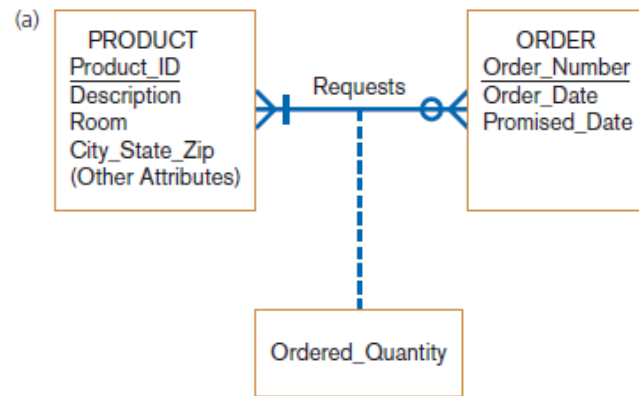
| <u>Order_Number</u> | Order_Date | Promised_Date | <u>Customer_ID</u> |
|---------------------|------------|---------------|--------------------|
| 57194 | 3/15/1X | 3/28/1X | 6390 |
| 63725 | 3/17/1X | 4/01/1X | 1273 |
| 80149 | 3/14/1X | 3/24/1X | 6390 |

(b) Relations



Binary and Higher-Degree M:N Relationships

- Create another relation and include primary keys of all relations as primary key of new relation



(b) ORDER

| <u>Order_Number</u> | Order_Date | Promised_Date |
|---------------------|------------|---------------|
| 61384 | 2/17/2014 | 3/01/2014 |
| 62009 | 2/13/2014 | 2/27/2014 |
| 62807 | 2/15/2014 | 3/01/2014 |

ORDER LINE

| <u>Order_Number</u> | <u>Product_ID</u> | Quantity_Ordered |
|---------------------|-------------------|------------------|
| 61384 | M128 | 2 |
| 61384 | A261 | 1 |

PRODUCT

| <u>Product_ID</u> | Description | Room | (Other Attributes) |
|-------------------|-------------|--------|--------------------|
| M128 | Bookcase | Study | – |
| A261 | Wall unit | Family | – |
| R149 | Cabinet | Study | – |

FIGURE 9-12

Representing an *M:N* relationship

(a) E-R diagram

(b) Relations



Unary Relationships

- **Unary 1:N Relationship**

- Is modeled as a relation
- Primary key of that relation is the same as for the entity type
- Foreign key is added to the relation that references the primary key values

- **Recursive foreign key:** a foreign key in a relation that references the primary key values of that same relation



Unary Relationships

■ **Unary M:N Relationship**

- Model as one relation, then
- Create a separate relation to represent the M:N relationship.
- The primary key of the new relation is a composite key of two attributes that both take their values from the same primary key.
- Any attribute associated with the relationship is included as a nonkey attribute in this new relation.

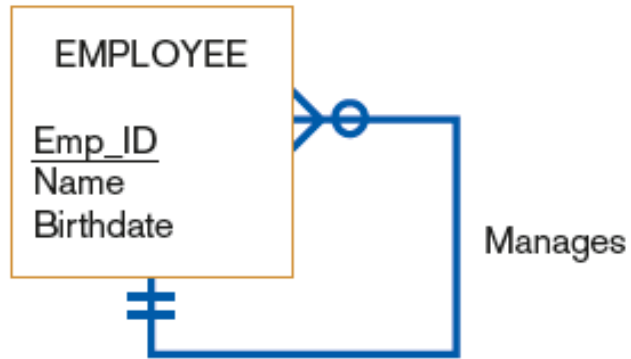
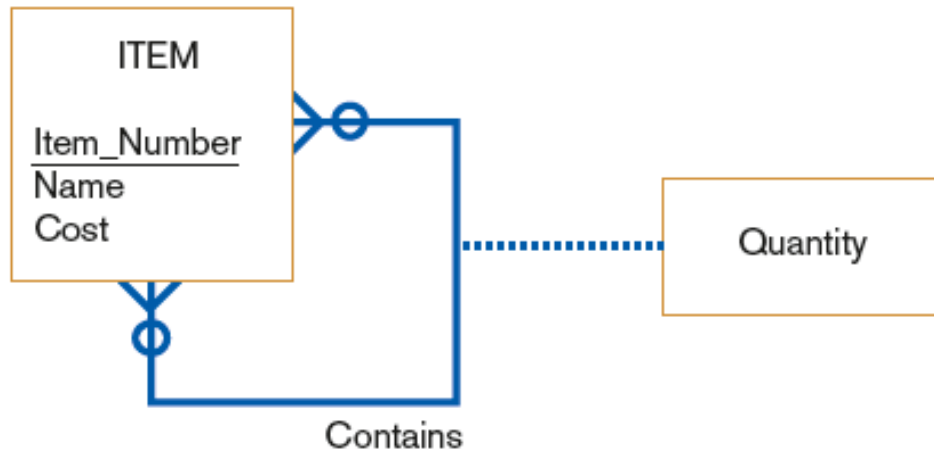


FIGURE 9-13
Two unary relationships

(a) EMPLOYEE with
Manages
relationship (1:N)



(b) Bill-of-materials
structure (M:N)

ITEM(Item_Number,Name,Cost)

ITEM-BILL(Item_Number,Component_Number,Quantity)



Merging Relations

- Purpose is to remove redundant relations
- The last step in logical database design
- Prior to physical file and database design



View Integration Problems

- Must understand the meaning of the data and be prepared to resolve any problems that arise in the process
- **Synonyms:** two different names used for the same attribute
 - When merging, get agreement from users on a single, standard name.



View Integration Problems (Cont.)

- **Homonyms:** a single attribute name that is used for two or more different attributes.
 - Resolved by creating a new name
- **Dependencies between nonkeys—** dependencies may be created as a result of view integration
 - To resolve, the new relation must be normalized



View Integration Problems (Cont.)

- **Class/Subclass** — relationships may be hidden in user views or relations
 - Resolved by creating a new name

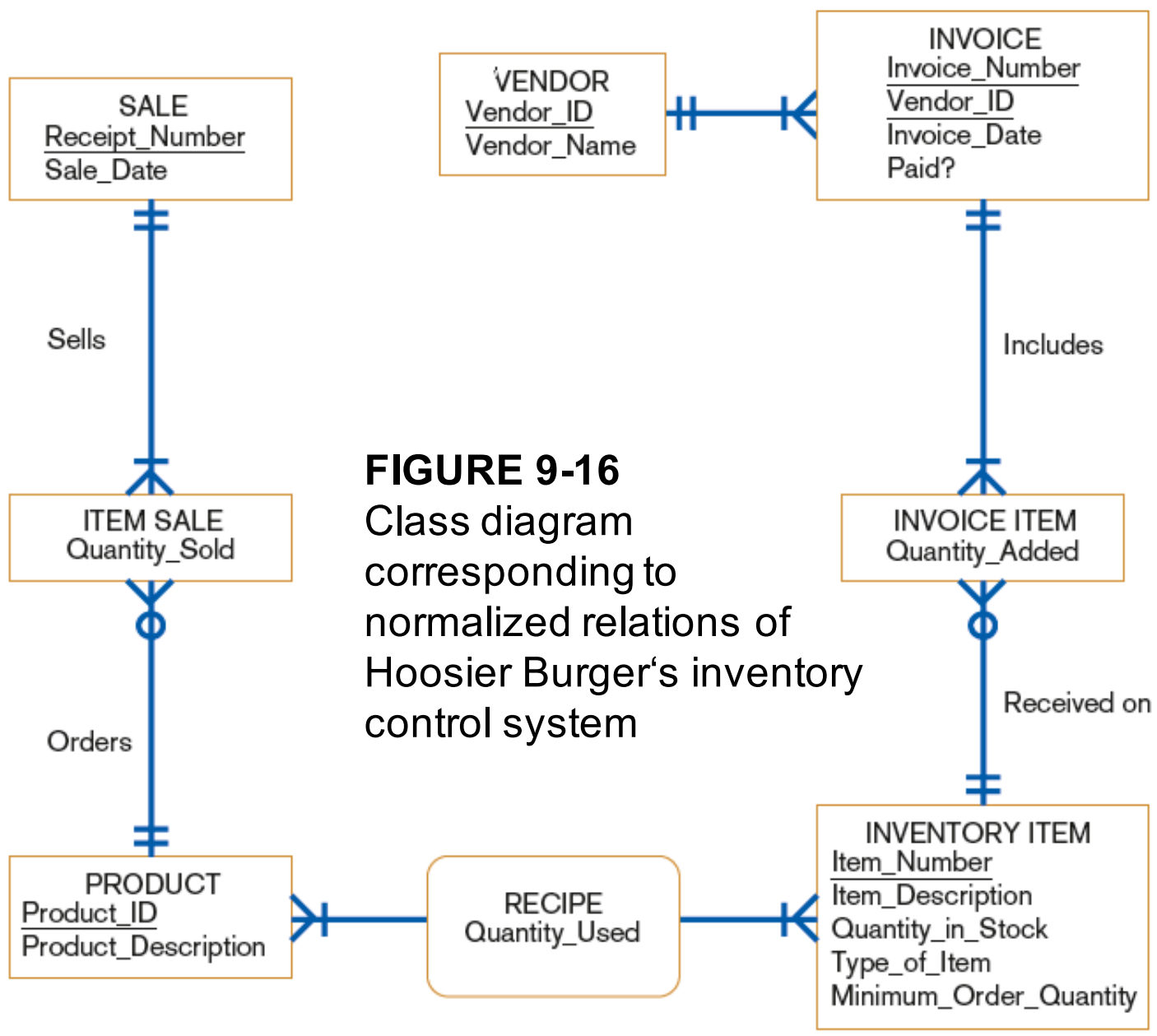
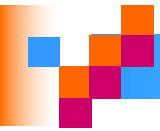


FIGURE 9-16 Class diagram corresponding to normalized relations of Hoosier Burger's inventory control system



Relations for Hoosier Burger

SALE(Receipt Number,Sale_Date)

PRODUCT(Product ID,Product_Description)

INVOICE(Vendor ID,Invoice Number,Invoice_Date,Paid?)

INVENTORY ITEM(Item Number,Item_Description,Quantity_in_Stock,
Minimum_Order_Quantity,Type_of_Item)

ITEM SALE(Receipt Number,Product ID,Quantity_Sold)

INVOICE ITEM(Vendor ID,Invoice Number,Item Number,Quantity_Added)

RECIPE(Product ID,Item Number,Quantity_Used)

VENDOR(Vendor ID,Vendor_Name)



Physical File and Database Design

- The following information is required:
 - Normalized relations, including volume estimates
 - Definitions of each attribute
 - Descriptions of where and when data are used, entered, retrieved, deleted, and updated (including frequencies)
 - Expectations or requirements for response time and data integrity
 - Descriptions of the technologies used for implementing the files and database



Designing Fields

- **Field:** the smallest unit of named application data recognized by system software
 - Attributes from relations will be represented as fields
- **Data Type:** a coding scheme recognized by system software for representing organizational data



Choosing Data Types

- Selecting a data type balances four objectives:
 - Minimize storage space.
 - Represent all possible values of the field.
 - Improve data integrity of the field.
 - Support all data manipulations desired on the field.



TABLE 9-2 Commonly Used Data Types in Oracle 10i

| Data Type | Description |
|-----------|--|
| VARCHAR2 | Variable-length character data with a maximum length of 4000 characters; you must enter a maximum field length (e.g., VARCHAR2(30) for a field with a maximum length of 30 characters). A value less than 30 characters will consume only the required space. |
| CHAR | Fixed-length character data with a maximum length of 255 characters; default length is 1 character (e.g., CHAR(5) for a field with a fixed length of five characters, capable of holding a value from 0 to 5 characters long). |
| LONG | Capable of storing up to two gigabytes of one variable-length character data field (e.g., to hold a medical instruction or a customer comment). |
| NUMBER | Positive and negative numbers in the range 10^{-130} to 10^{126} ; can specify the precision (total number of digits to the left and right of the decimal point) and the scale (the number of digits to the right of the decimal point) (e.g., NUMBER(5) specifies an integer field with a maximum of 5 digits and NUMBER(5, 2) specifies a field with no more than five digits and exactly two digits to the right of the decimal point). |
| DATE | Any date from January 1, 4712 B.C. to December 31, 4712 A.D.; date stores the century, year, month, day, hour, minute, and second. |
| BLOB | Binary large object, capable of storing up to four gigabytes of binary data (e.g., a photograph or sound clip). |



Calculated Fields

- **Calculated (or computed or derived) field:** a field that can be derived from other database fields
- It is common for an attribute to be mathematically related to other data.
- The calculate value is either stored or computed when it is requested.



Controlling Data Integrity

- **Default Value:** a value a field will assume unless an explicit value is entered for that field
- **Range Control:** limits range of values that can be entered into field
 - Both numeric and alphanumeric data
- **Referential Integrity:** an integrity constraint specifying that the value (or existence) of an attribute in one relation depends on the value (or existence) of the same attribute in another relation
- **Null Value:** a special field value, distinct from zero, blank, or any other value, that indicates that the value for the field is missing or otherwise unknown



Designing Physical Tables

- Relational database is a set of related tables.
- **Physical Table:** a named set of rows and columns that specifies the fields in each row of the table
- **Denormalization:** the process of splitting or combining normalized relations into physical tables based on affinity of use of rows and fields
- Denormalization optimizes certain data processing activities at the expense of others.



Designing Physical Tables (Cont.)

- Three types of table partitioning:
 - *Range partitioning*: partitions are defined by nonoverlapping ranges of values for a specified attribute
 - *Hash partitioning*: a table row is assigned to a partition by an algorithm and then maps the specified attribute value to a partition
 - *Composite partitioning*: combines range and hash partitioning by first segregating data by ranges on the designated attribute, and then within each of these partitions



Designing Physical Tables (Cont.)

- Various forms of denormalization, which involves combining data from several normalized tables, can be done.
 - No hard-and-fast rules for deciding
- Three common situations where denormalization may be used:
 - *Two entities with a one-to-one relationship*
 - *A many-to-many relationship (associative entity) with nonkey attributes*
 - *Reference data*



File Organizations

- **File organization:** a technique for physically arranging the records of a file
- **Physical file:** a named set of table rows stored in a contiguous section of secondary memory

File Organizations (Cont.)

TABLE 9-3 Comparative Features of Sequential, Indexed, and Hashed File Organizations

| Factor | File Organization | | |
|-------------------------------------|---|--|---|
| | Sequential | Indexed | Hashed |
| Storage space | No wasted space | No wasted space for data, but extra space for index | Extra space may be needed to allow for addition and deletion of records |
| Sequential retrieval on primary key | Very fast | Moderately fast | Impractical |
| Random retrieval on primary key | Impractical | Moderately fast | Very fast |
| Multiple key retrieval | Possible, but requires scanning whole file | Very fast with multiple indexes | Not possible |
| Deleting rows | Can create wasted space or require reorganizing | If space can be dynamically allocated, this is easy, but requires maintenance of indexes | Very easy |
| Adding rows | Requires rewriting file | If space can be dynamically allocated, this is easy, but requires maintenance of indexes | Very easy, except multiple keys with same address require extra work |
| Updating rows | Usually requires rewriting file | Easy, but requires maintenance of indexes | Very easy |



File Organizations (Cont.)

- **Sequential file organization:** a file organization in which rows in a file are stored in sequence according to a primary key value
- **Hashed file organization:** a file organization in which the address for each row is determined using an algorithm
- **Pointer:** a field of data that can be used to locate a related field or row of data



Arranging Table Rows

- Objectives for choosing file organization
 - Fast data retrieval
 - High throughput for processing transactions
 - Efficient use of storage space
 - Protection from failures or data loss
 - Minimizing need for reorganization
 - Accommodating growth
 - Security from unauthorized use



Indexed File Organization

- **Indexed file organization:** a file organization in which rows are stored either sequentially or nonsequentially, and an index is created that allows software to locate individual rows
- **Index:** a table used to determine the location of rows in a file that satisfy some condition
- **Secondary keys:** one or a combination of fields for which more than one row may have the same combination of values



Indexed File Organization (Cont.)

- Main disadvantages:

- Extra space required to store the indexes
- Extra time necessary to access and maintain indexes

- Main advantage:

- Allows for both random and sequential processing

- Guidelines for choosing indexes

- Specify a unique index for the primary key of each table.
- Specify an index for foreign keys.
- Specify an index for nonkey fields that are referenced in qualification, sorting and grouping commands for the purpose of retrieving data.



Designing Controls for Files

- Two of the goals of physical table design are *protection from failure or data loss* and *security from unauthorized use*.
- These goals are achieved primarily by implementing controls on each file.
- Two other important types of controls *address file backup and security*.



Designing Controls for Files (Cont.)

- Techniques for file restoration include:
 - Periodically making a backup copy of a file.
 - Storing a copy of each change to a file in a transaction log or audit trail.
 - Storing a copy of each row before or after it is changed.
- Means of building data security into a file include:
 - Coding, or encrypting, the data in the file.
 - Requiring data file users to identify themselves by entering user names and passwords.
 - Prohibiting users from directly manipulating any data in the file by forcing users to work with a copy (real or virtual).



Physical Database Design for Hoosier Burger

- The following decisions need to be made:
 - Decide to create one or more fields for each attribute and determine a data type for each field.
 - For each field, decide if it is calculated; needs to be coded or compressed; must have a default value or picture; or must have range, referential integrity, or null value controls.
 - For each relation, decide if it should be denormalized to achieve desired processing efficiencies.
 - Choose a file organization for each physical file.
 - Select suitable controls for each file and the database.



Electronic Commerce Application: Designing Databases

- Designing databases for Pine Valley Furniture's WebStore
 - Review the conceptual model (E-R diagram).
 - Examine the lists of attributes for each entity.
 - Complete the database design.
 - Share all design information with project team to be turned into a working database during implementation.



Summary

- In this chapter you learned how to:
 - ✓ Concisely define each of the following key database design terms: relation, primary key, normalization, functional dependency, foreign key, referential integrity, field, data type, null value, denormalization, file organization, index, and secondary key.
 - ✓ Explain the role of designing databases in the analysis and design of an information system.
 - ✓ Transform an entity-relationship (E-R) diagram into an equivalent set of well-structured (normalized) relations.



Summary (Cont.)

- ✓ Merge normalized relations from separate user views into a consolidated set of well-structured relations.
- ✓ Choose storage formats for fields in database tables.
- ✓ Translate well-structured relations into efficient database tables.
- ✓ Explain when to use different types of file organizations to store computer files.
- ✓ Describe the purpose of indexes and the important considerations in selecting attributes to be indexed.



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.

Copyright © 2014 Pearson Education, Inc.
Publishing as Prentice Hall