Modern Systems Analysis and Design

Eighth Edition, Global Edition

Joseph S. Valacich
Joey F. George

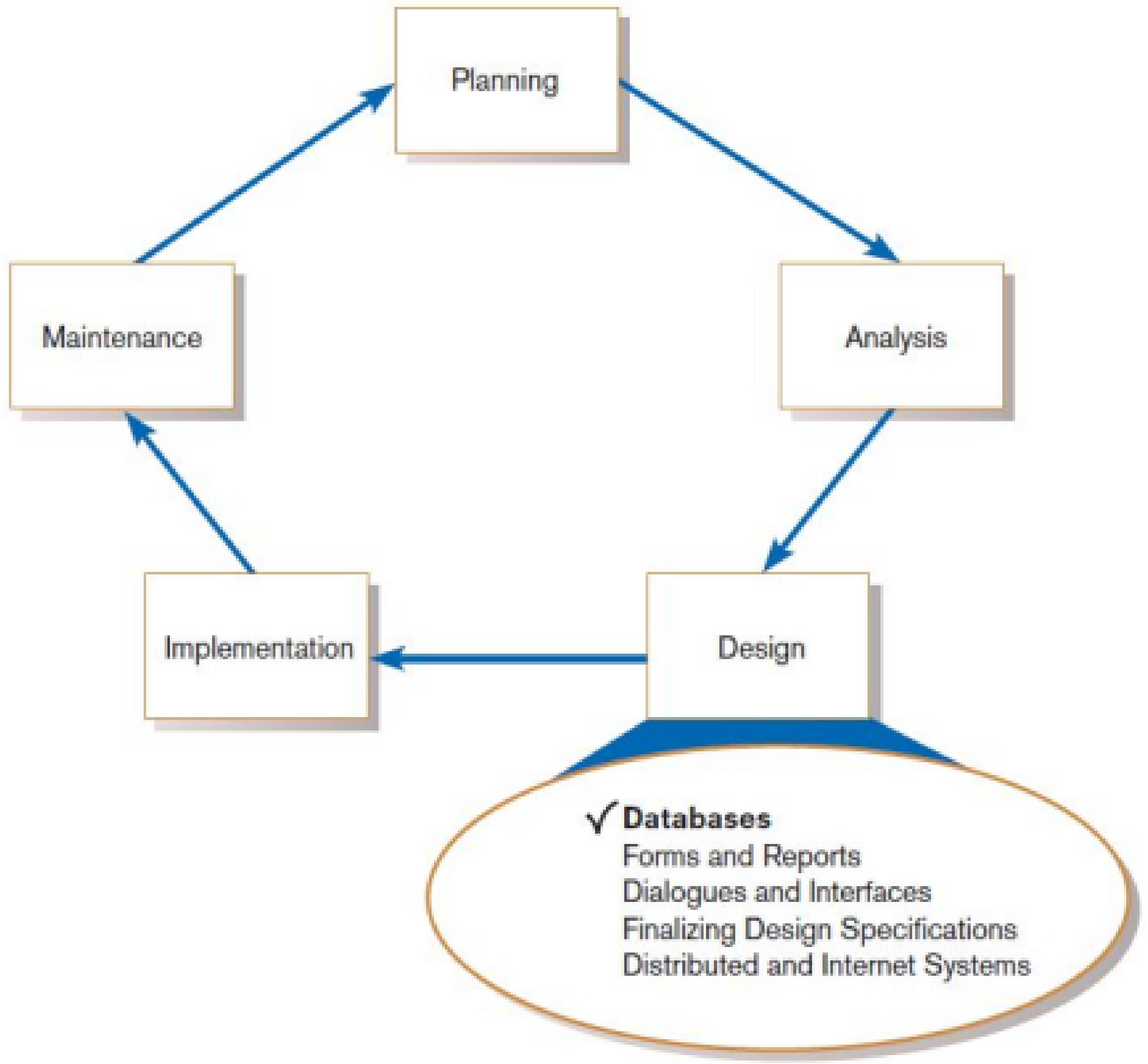
Designing Databases

Learning Objectives

- ✔ Describe the database design process, its outcomes, and the relational database model.
- ✔ Describe normalization and the rules for second and third normal form.
- ✔ Transform an entity-relationship (E-R) diagram into an equivalent set of well-structured (normalized) relations.
- ✔ Merge normalized relations from separate user views into a consolidated set of well-structured relations.

Learning Objectives (Cont.)

- ✓ Describe physical database design concepts: ✓ 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

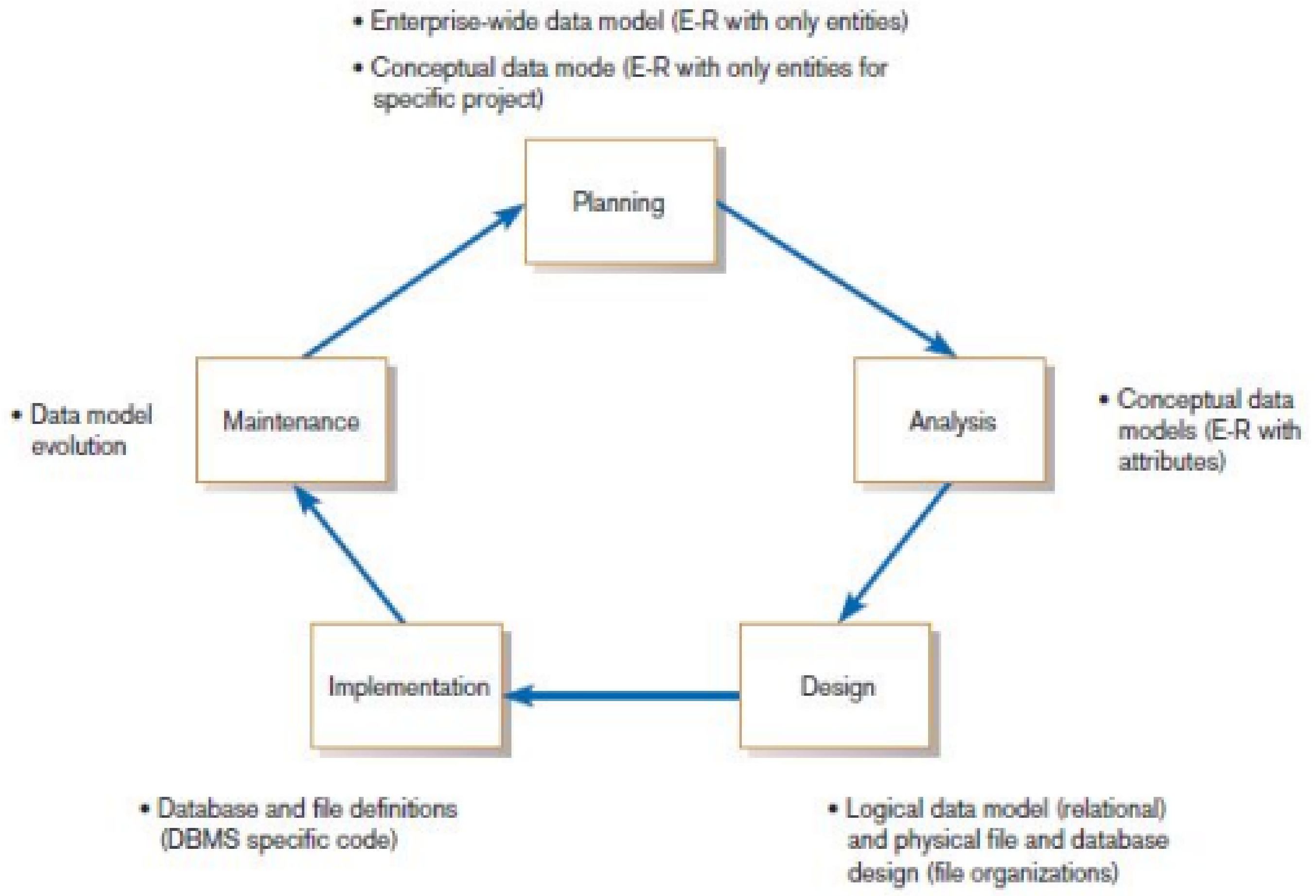
Database Design

- 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


• Enterprise-wide data model (E-R with only entities)
• Conceptual data mode (E-R with only entities for specific project)
Planning
Analysis
• Conceptual data models (E-R with attributes)
Design
• Logical data model (relational) and physical file and database design (file organizations)
Implementation
• Database and file definitions (DBMS specific code)
Maintenance
• Data model evolution

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

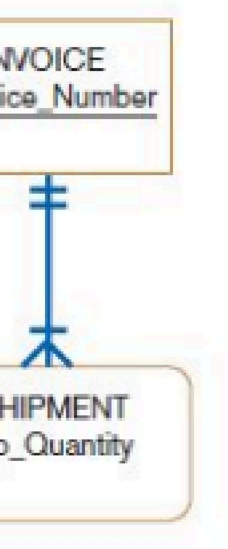
- Logical database design
  - Must account for every data element on a system input or output
    - Normalized relations are the primary deliverable.
- Physical database design
  - Converts relations into database tables
    - Programmers and database analysts code the definitions of the database.
    - Written in Structured Query Language (SQL)

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

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



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.)

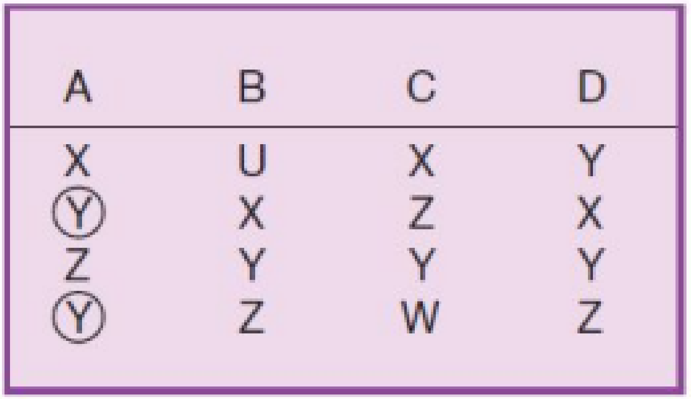
- First Normal Form (1NF)
  - Unique rows, no multivalued attributes
  - All relations are in 1NF
- Second Normal Form (2NF)
  - Each nonprimary key attribute is identified by the whole primary key (called full functional dependency)
- Third Normal Form (3NF)
  - Nonprimary key attributes do not depend 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.
- Rather, the Functional dependence of B on A means that there can be only one value of B for each value of A. An attribute may be functionally dependent on two (or more) attributes rather than on a single attribute.



A	B	C	D
X	U	X	Y
Ⓨ	X	Z	X
Z	Y	Y	Y
Ⓨ	Z	W	Z

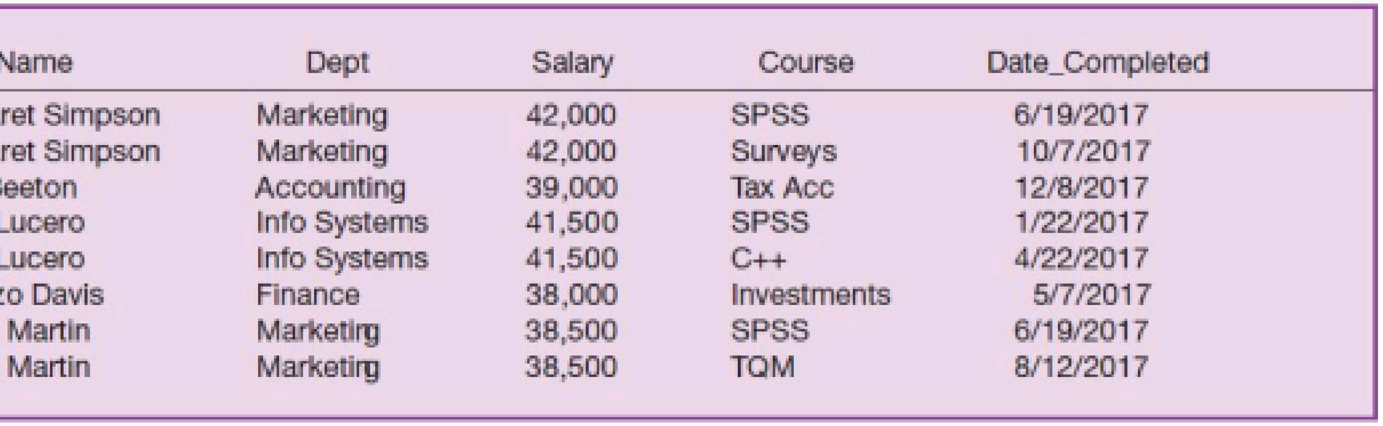
Second Normal Form (2NF)

- 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

ıtes.

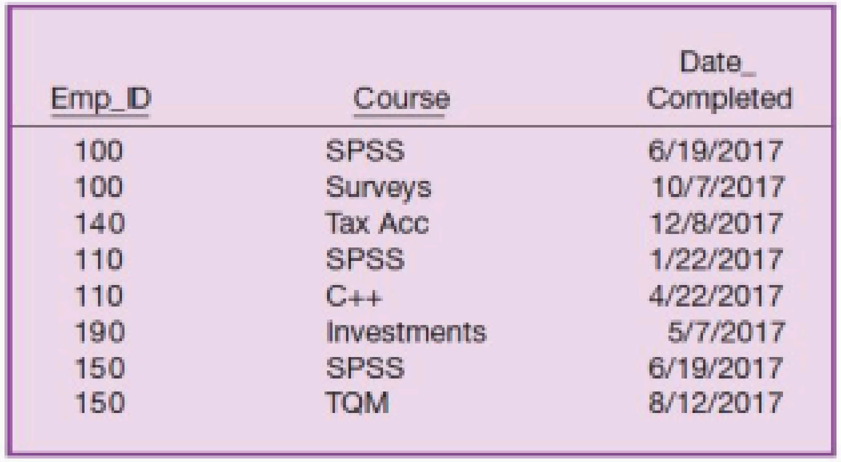
the new



Name	Dept	Salary	Course	Date_Completed
ret Simpson	Marketing	42,000	SPSS	6/19/2017
ret Simpson	Marketing	42,000	Surveys	10/7/2017
eeton	Accounting	39,000	Tax Acc	12/8/2017
Lucero	Info Systems	41,500	SPSS	1/22/2017
Lucero	Info Systems	41,500	C++	4/22/2017
zo Davis	Finance	38,000	Investments	5/7/2017
Martin	Marketing	38,500	SPSS	6/19/2017
Martin	Marketing	38,500	TQM	8/12/2017

-18

EMP COURSE



Emp_ID	Course	Date_ Completed
100	SPSS	6/19/2017
100	Surveys	10/7/2017
140	Tax Acc	12/8/2017
110	SPSS	1/22/2017
110	C++	4/22/2017
190	Investments	5/7/2017
150	SPSS	6/19/2017
150	TQM	8/12/2017

Emp_ID

Name,Dept,Salary (partial dependency) **Emp_ID,Course**
Date_Completed (complete depencency)

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.

FIGURE 9-9
Removing transitive dependencies
(a) Relation with transitive dependency

Customer_ID
Customer_Name,Salesperson,Region

Salesperson Region

(b) Relation in 3NF

Region

Third Normal Form (3NF) (Cont.)

- **Foreign Key:** an attribute that appears as a nonprimary key attribute (or part of a primary key) in one relation and as a primary key attribute 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 1:1Relationships

- ■ 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 1:1Relationships (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

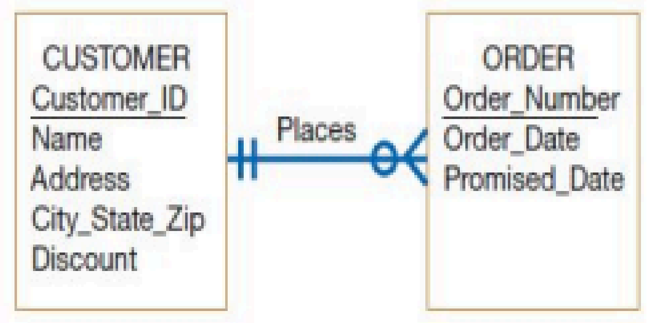
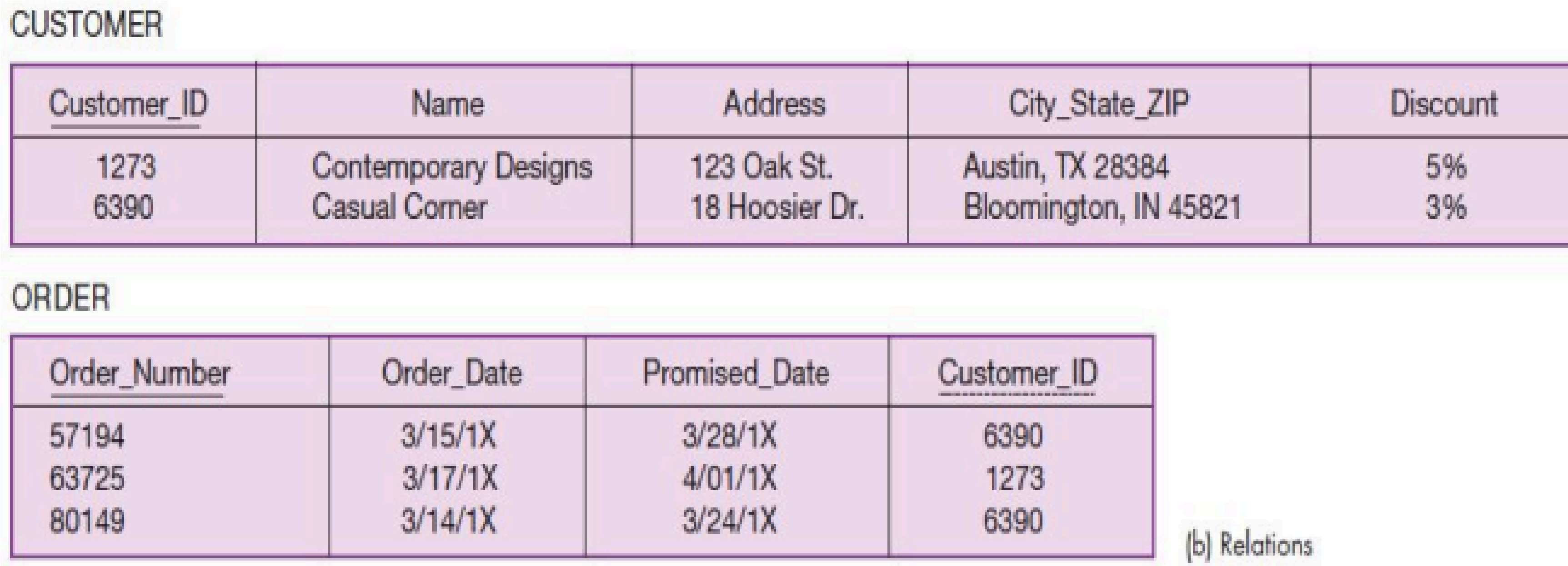


FIGURE 9-11
Representing a 1:N relationship
(a) E-R diagram



CUSTOMER

Customer_ID	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

Order_Number	Order_Date	Promised_Date	Customer_ID
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

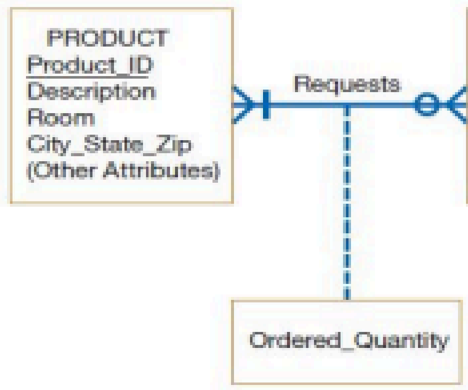
Binary and Higher-Degree M:N

Relationships

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

FIGURE 9-12
Representing an *M:N* relationship
(a) E-R diagram

(a)



(b) Relations

(b) ORDER

Order_Number	Order_Date	Promised_Date
61384	2/17/2014	3/01/2017
62009	2/13/2014	2/27/2017
62807	2/15/2014	3/01/2017

ORDER LINE

Order_Number	Product_ID	Quantity_ Ordered
61384	M128	2
61384	A261	1

PRODUCT

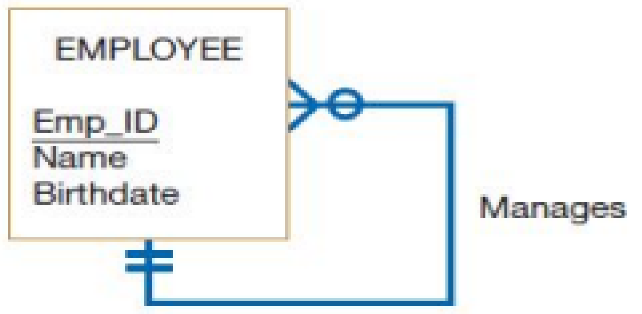
Product_ID	Description	Room	(Other Attributes)
M128	Bookcase	Study	—
A261	Wall unit	Family	—
R149	Cabinet	Study	—

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

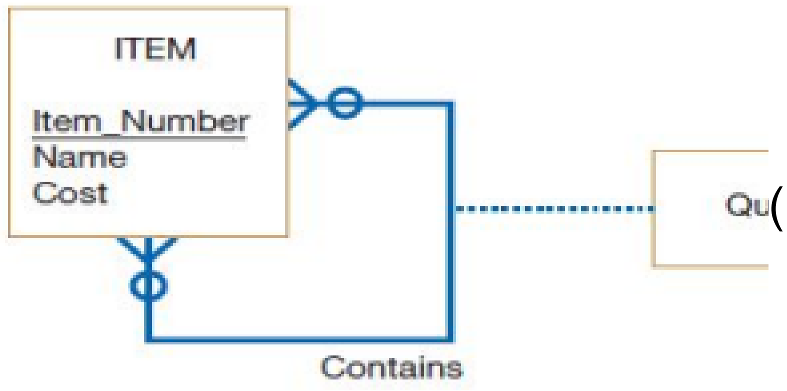
- **Unary M: N Relationship**
  - Model as one relation, then
  - Create a separate relation to representing the M: N relationship.
    - The primary key of the new relation is a composite key of two attributes that both take their values from the same primary key.
  - Any attribute associated with the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



EMPLOYEE (Emp_ID, Name, Birthdate, Manager_ID)

FIGURE 9-13
Two unary relationships

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



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

ITEM(<u>Item_Number</u>,Name,Cost)
ITEM-BILL(<u>Item_Number,Component_Number</u>,Quantity)

Merging Relations

- Purpose is to remove redundant relations
- The last step in logical database design
- Redundant relations could come about due to multiple E-R diagrams and/or user interfaces
- Prior to physical file and database design
- Example: given two relations:
  - EMPLOYEE1(<u>Emp_ID</u>,Name,Address,Phone)

- EMPLOYEE2(<u>Emp_ID</u>,Name,Address,Jobcode,Number_of_Years)

- You can merge them together:
  - EMPLOYEE(<u>Emp_ID</u>,Name,Address,Phone,Jobcode,Number_of_Years)

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
  - An attribute from a logical database model may be represented by several fields. For example, a student name attribute in a normalized student relation might be

represented as three fields: last name, first name, and middle initial.

- **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.

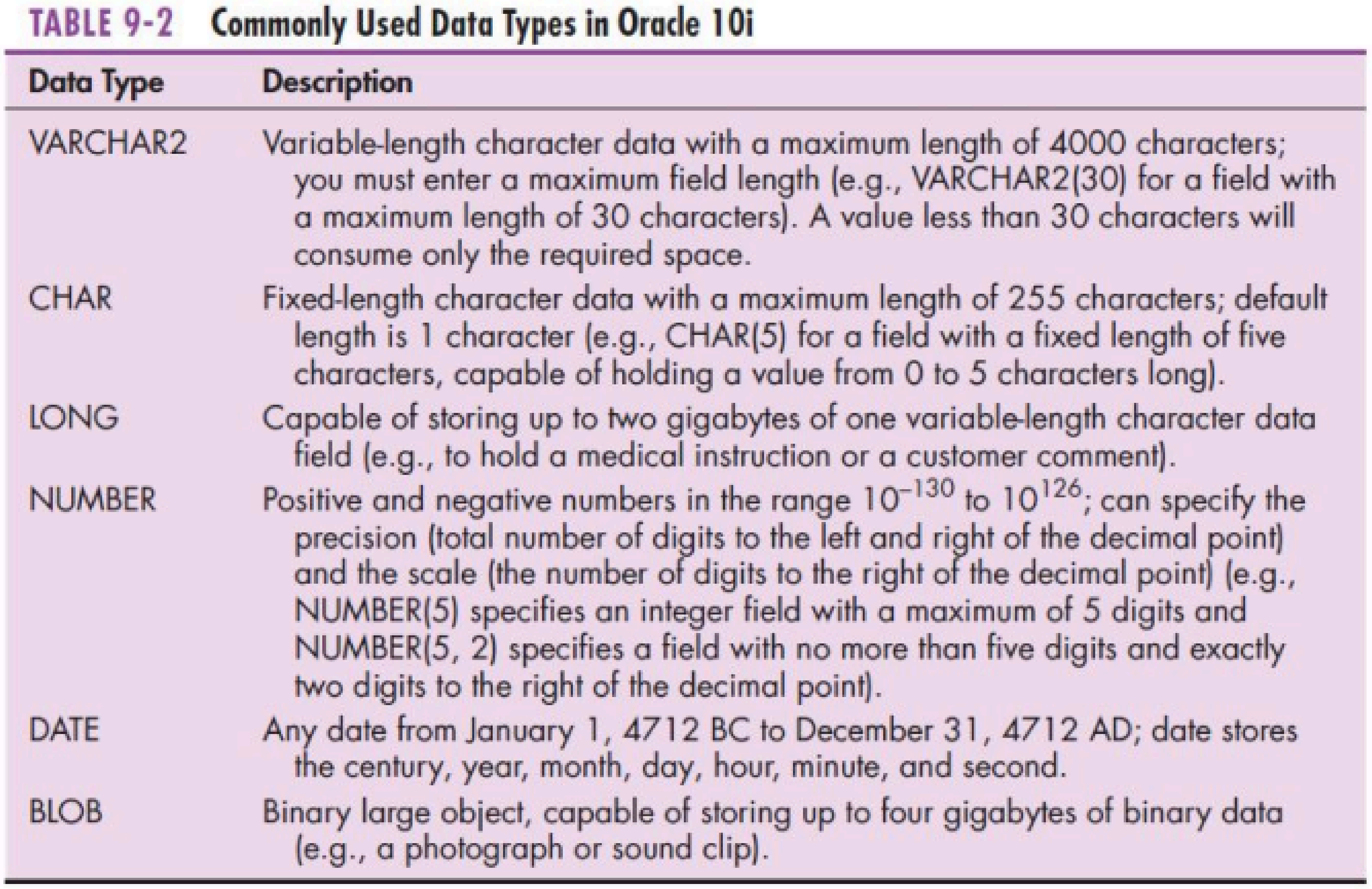


TABLE 9-2 Commonly Used Data Types in Oracle 10i

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 BC to December 31, 4712 AD; 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.

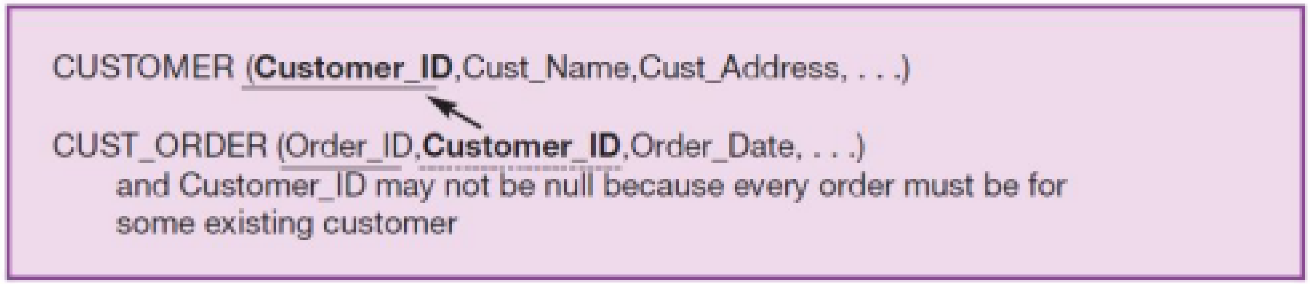
Controlling Data Integrity

- **Default Value**: a value a field will assume unless an explicit value is entered for that field
- **Range Control**: limits the range of values that can be entered into the field
  - Both numeric and alphanumeric data

- **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

Controlling Data Integrity

- **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



CUSTOMER (**Customer_ID**,Cust_Name,Cust_Address, . . .)

CUST_ORDER (Order_ID,**Customer_ID**,Order_Date, . . .)
and Customer_ID may not be null because every order must be for some existing customer

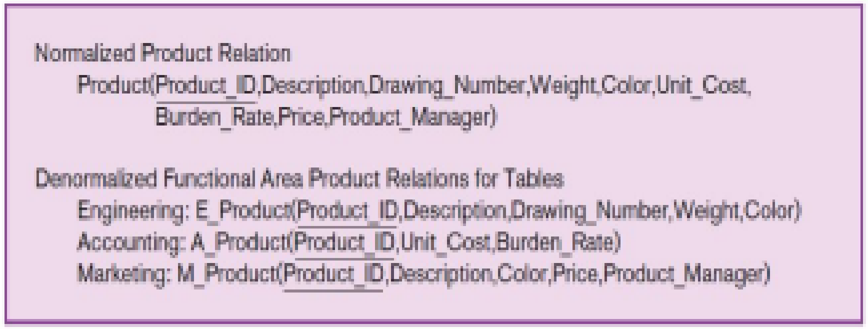
The values for the foreign keyCustomer_ID field within a customer order must be limited to the set of Customer_ID values from the CUSTOMER relation; we would not want to accept an order for a nonexisting or unknown customer.

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.
- The design of a physical table has two goals different from those of normalization: efficient use of secondary storage and data processing speed.

Designing Physical Tables (Cont.)

Fig: Denormalization by columns



Normalized Product Relation

Product(Product_ID,Description,Drawing_Number,Weight,Color,Unit_Cost, Burden_Rate,Price,Product_Manager)

Denormalized Functional Area Product Relations for Tables

Engineering: E_Product(Product_ID,Description,Drawing_Number,Weight,Color)

Accounting: A_Product(Product_ID,Unit_Cost,Burden_Rate)

Marketing: M_Product(Product_ID,Description,Color,Price,Product_Manager)

Fig: Denormalization by rows

Designing Physical Tables (Cont.)

- **Partitioning**: splitting a table into different physical files, perhaps stored on different disks or computer.
- Helps speed up system performance.
- 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

File Organizations

- **Physical file**: a named set of table rows stored in a contiguous section of secondary memory. A file contains rows and columns from one or more tables, as produced from denormalization.
- **File organization**: a technique for

physically arranging the records of a file

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

File Organizations (Cont.)

- **Three common file organizations:**
- **Sequential**: rows are stored in sequence according to a primary key value. To locate a particular row, a program must normally scan the file from the beginning until the desired row is located.
- **Hashed file organization**: rows are usually stored nonsequentially; the address for each row is determined

using an algorithm that converts a primary key value into a row address.

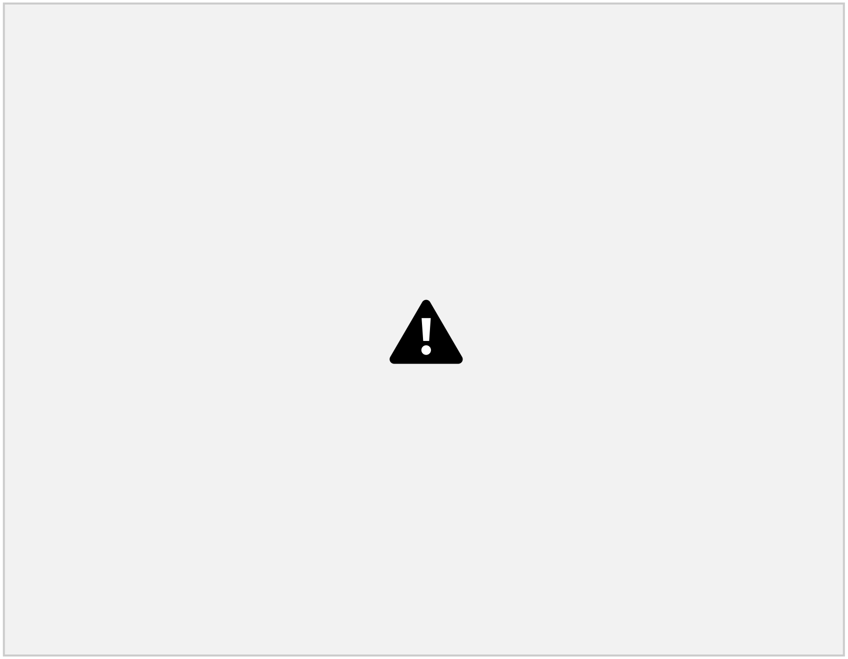
File Organizations (Cont.)

- **Three common file organizations:**
- **Indexed:** rows can be stored sequentially or nonsequentially; an index allows quick access to rows.
  - One of the most powerful capabilities of indexed file organizations is the ability to create multiple indexes, similar to the title, author, and subject indexes in a library.
  - The main disadvantages to indexed file organizations are the extra space required to store the indexes and the extra time necessary to

access and maintain indexes.

Figure 9-20 Comparison of file organizations

(a)



Sequential

(b) Indexed

(c) Hashed

File Organizations (Cont.)

Summary

- In this chapter you learned how to: ✓
  Describe the database design process, its outcomes, and the relational database model.
  - ✓ Describe normalization and the rules for second and third normal form.
  - ✓ Transform an entity-relationship (E-R) diagram into an equivalent set of well-structured (normalized) relations.
  - ✓ Merge normalized relations from separate user views into a consolidated set of well-structured relations.

Summary (Cont.)

- ✓ Describe physical database design concepts: ✓ 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.