11e

**Database Systems**

**Design, Implementation, and Management**

Chapter 3

The Relational Database

Model

Learning Objectives

In this chapter, one will learn:

That the relational database model offers a logical view of data

About the relational model’s basic component:

relations

That relations are logical constructs composed of rows

(tuples) and columns (attributes)

That relations are implemented as tables in a relational

DBMS

Learning Objectives

In this chapter, one will learn:

About relational database operators, the data dictionary, and the system catalog

How data redundancy is handled in the relational database model

Why indexing is important

A Logical View of Data

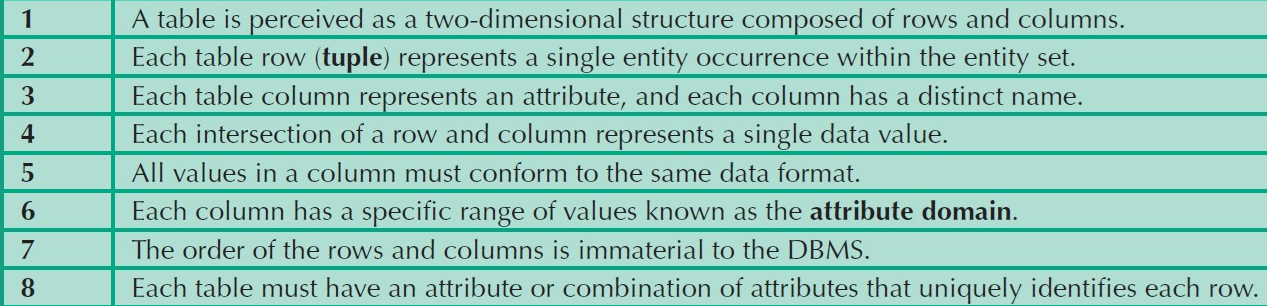
Relational database model enables logical representation of the data and its relationships

Logical simplicity yields simple and effective database design methodologies

Facilitated by the creation of data relationships based on a logical construct called a relation

Table 3.1 - Characteristics of a Relational

Table



Keys

Consist of one or more attributes that determine other attributes

Used to:

Ensure that each row in a table is uniquely identifiable

Establish relationships among tables and to ensure the integrity of the data

**Primary key (PK)**: Attribute or combination of attributes that uniquely identifies any given row

Determination

State in which knowing the value of one attribute makes it possible to determine the value of another

Is the basis for establishing the role of a key

Based on the relationships among the attributes

Dependencies

**Functional dependence**: Value of one or more attributes determines the value of one or more other attributes

**Determinant**: Attribute whose value determines another

**Dependent**: Attribute whose value is determined by the other attribute

**Full functional dependence**: Entire collection of attributes in the determinant is necessary for the relationship

Types of Keys

**Composite key**: Key that is composed of more than one attribute

**Key attribute**: Attribute that is a part of a key

**Entity integrity**: Condition in which each row in the table has its own unique identity

All of the values in the primary key must be unique

No key attribute in the primary key can contain a null

Types of Keys

**Null**: Absence of any data value that could represent:

An unknown attribute value

A known, but missing, attribute value

A inapplicable condition

**Referential integrity**: Every reference to an entity instance by another entity instance is valid

Table 3.3 - Relational Database Keys

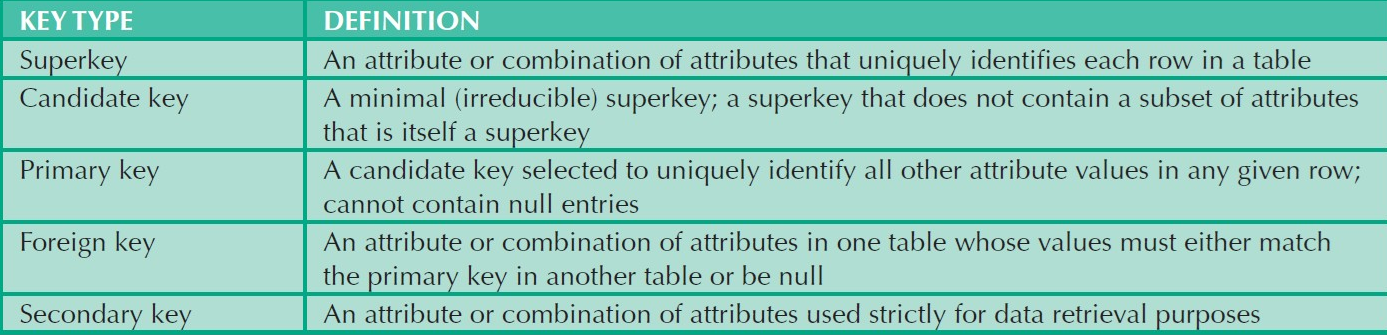
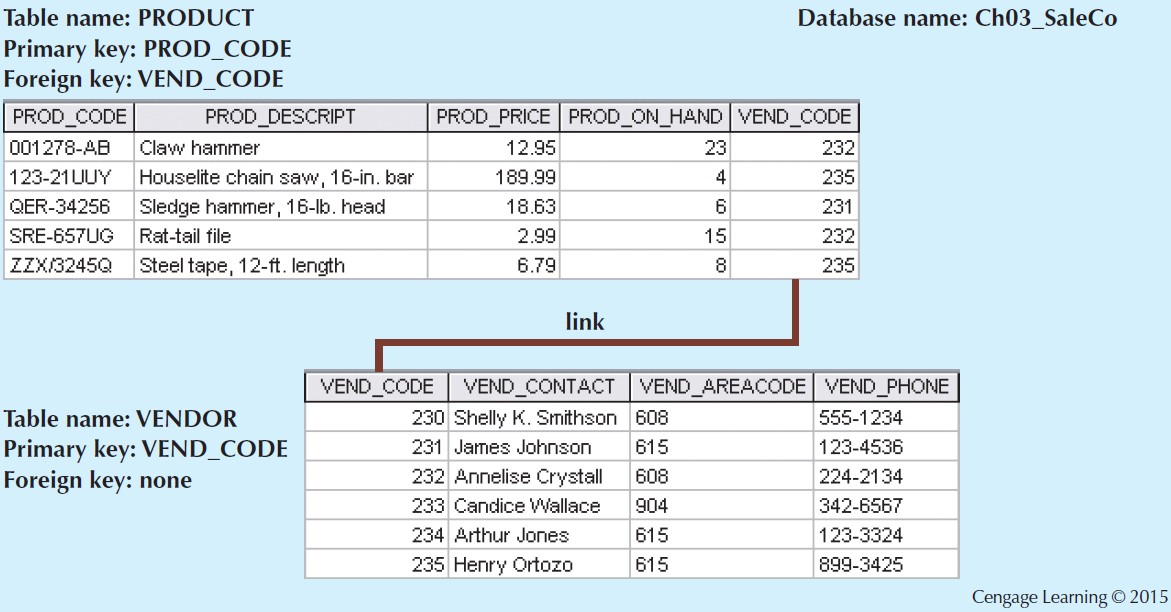


Figure 3.2 - An Example of a Simple

Relational Database



Integrity Rules

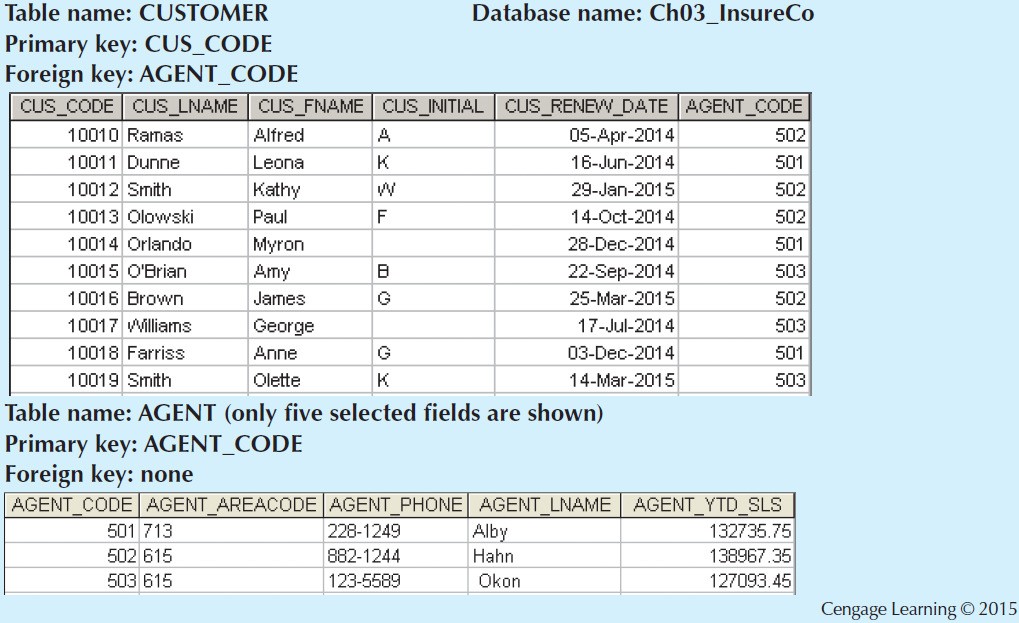
|  |
| --- |
| **Entity Integrity Description** |
| Requirement All primary key entries are unique, and no part of a primary key may be null |
| Purpose Each row will have a unique identity, and foreign key values can properly  reference primary key values |
| Example No invoice can have a duplicate number, nor it can be null |

Integrity Rules

|  |
| --- |
| **Entity Integrity Description** |
| Requirement A foreign key may have either a null entry or a entry that matches a primary key value in a table to which it is related |
| Purpose It is possible for an attribute not to have a corresponding value but it is impossible  to have an invalid entry  It is impossible to delete row in a table whose primary keys has mandatory matching foreign key values in another table |
| Example It is impossible to have invalid sales representative number |

Figure 3.3 - An Illustration of Integrity

Rules



Ways to Handle Nulls

**Flags**: Special codes used to indicate the absence of some value

NOT NULL constraint - Placed on a column to ensure that every row in the table has a value for that column

UNIQUE constraint - Restriction placed on a column to ensure that no duplicate values exist for that column

Relational Algebra

Theoretical way of manipulating table contents using relational operators

**Relvar**: Variable that holds a relation

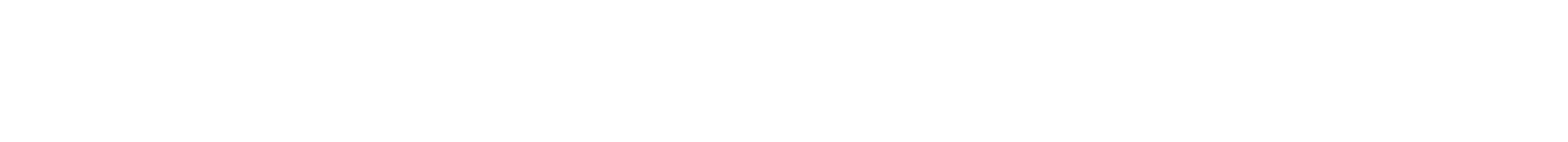
Heading contains the names of the attributes and the body contains the relation

Relational operators have the property of closure

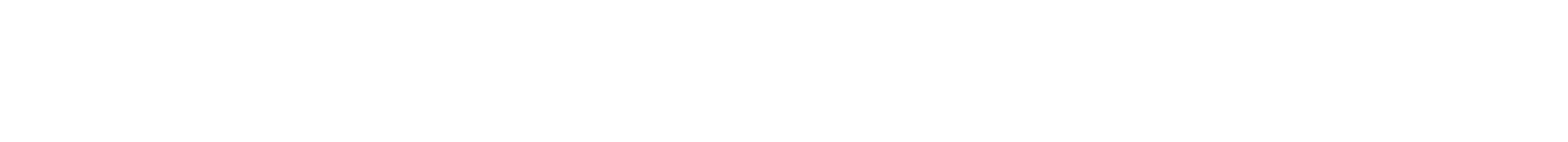
**Closure**: Use of relational algebra operators on existing relations produces new relations

Relational Set Operators

• Unary operator that yields a horizontal subset of a table

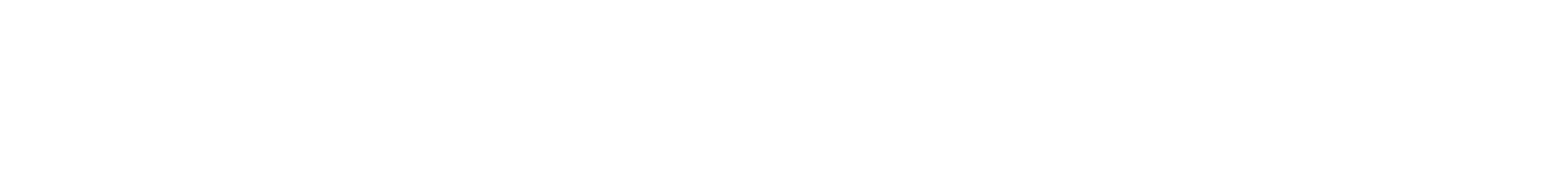
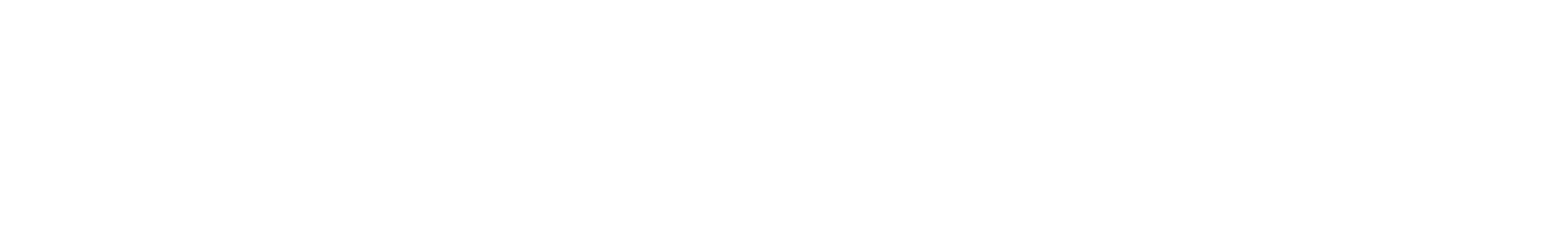


• Unary operator that yields a vertical subset of a table



• Combines all rows from two tables, excluding duplicate rows

• **Union-compatible**: Tables share the same number of columns, and their corresponding columns share compatible domains



• Yields only the rows that appear in both tables

• Tables must be union-compatible to yield valid results

Figure 3.4 - Select

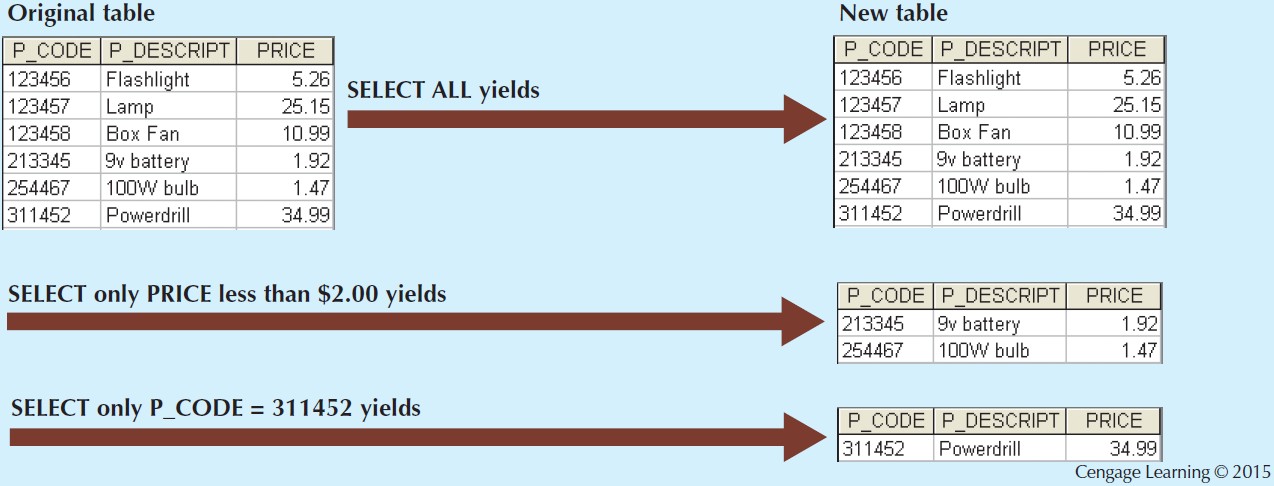


Figure 3.5 - Project

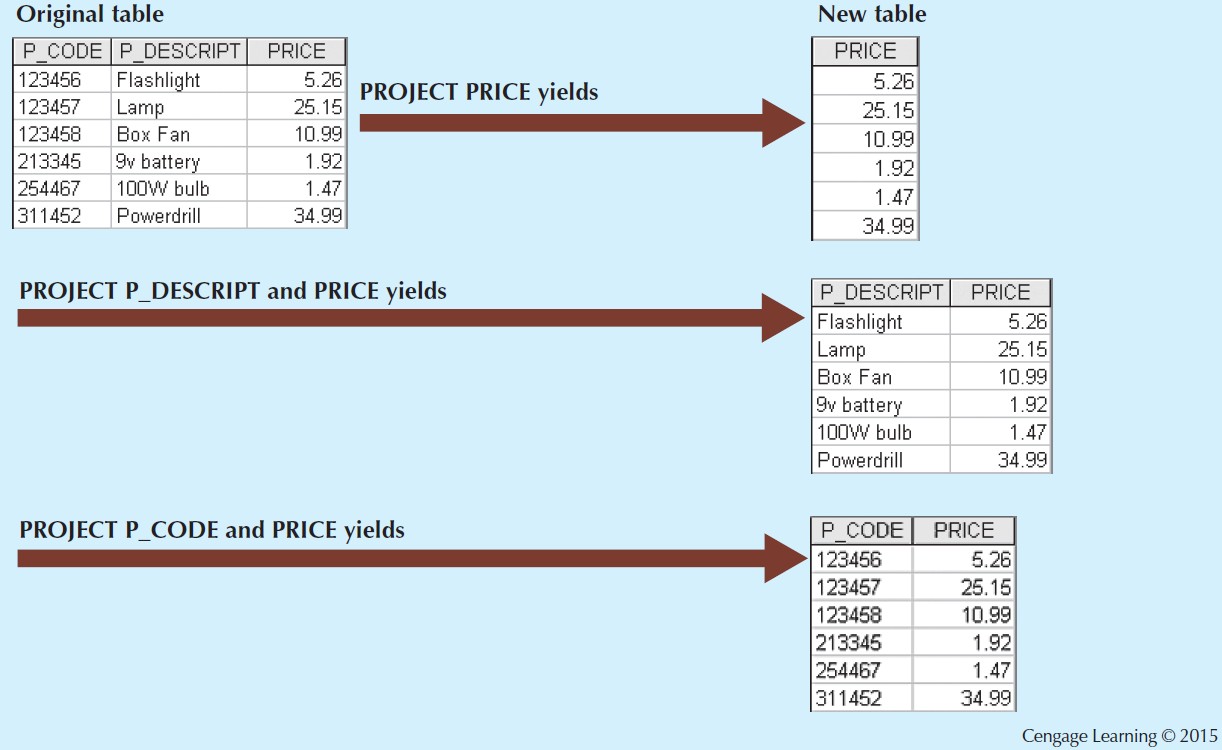
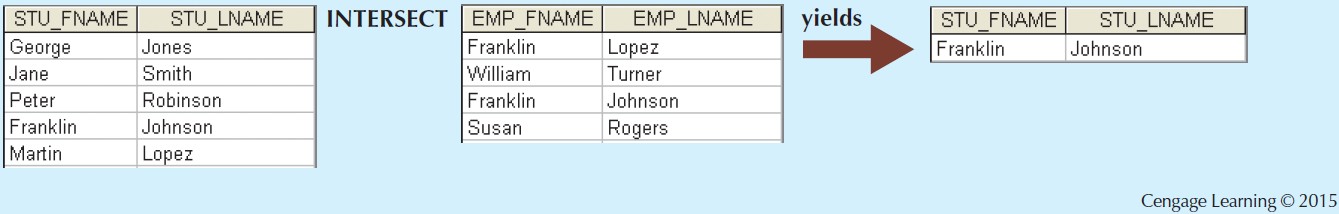


Figure 3.6 - Union



Figure 3.7 - Intersect



Relational Set Operators

**Difference**

Yields all rows in one table that are not found in the other table

Tables must be union-compatible to yield valid results

**Product**

Yields all possible pairs of rows from two tables

Relational Set Operators

**Join**

Allows information to be intelligently combined from two or more tables

**Divide**

Uses one 2-column table as the dividend and one single-column table as the divisor

Output is a single column that contains all values from the second column of the dividend that are associated with every row in the divisor

Types of Joins

**Natural join**: Links tables by selecting only the rows with common values in their common attributes

**Join columns**: Common columns

**Equijoin**: Links tables on the basis of an equality condition that compares specified columns of each table

**Theta join**: Extension of natural join, denoted by adding a theta subscript after the JOIN symbol

Types of Joins

**Inner join**: Only returns matched records from the tables that are being joined

**Outer join**: Matched pairs are retained and unmatched values in the other table are left null

**Left outer join**: Yields all of the rows in the first table, including those that do not have a matching value in

the second table

**Right outer join**: Yields all of the rows in the second table, including those that do not have matching values in the first table

Figure 3.8 - Difference

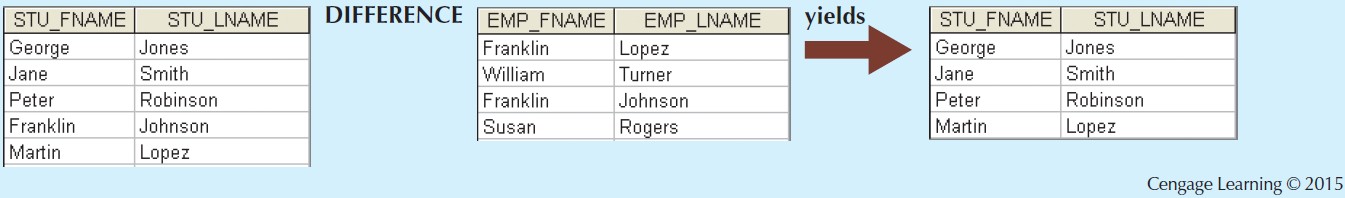


Figure 3.9 - Product

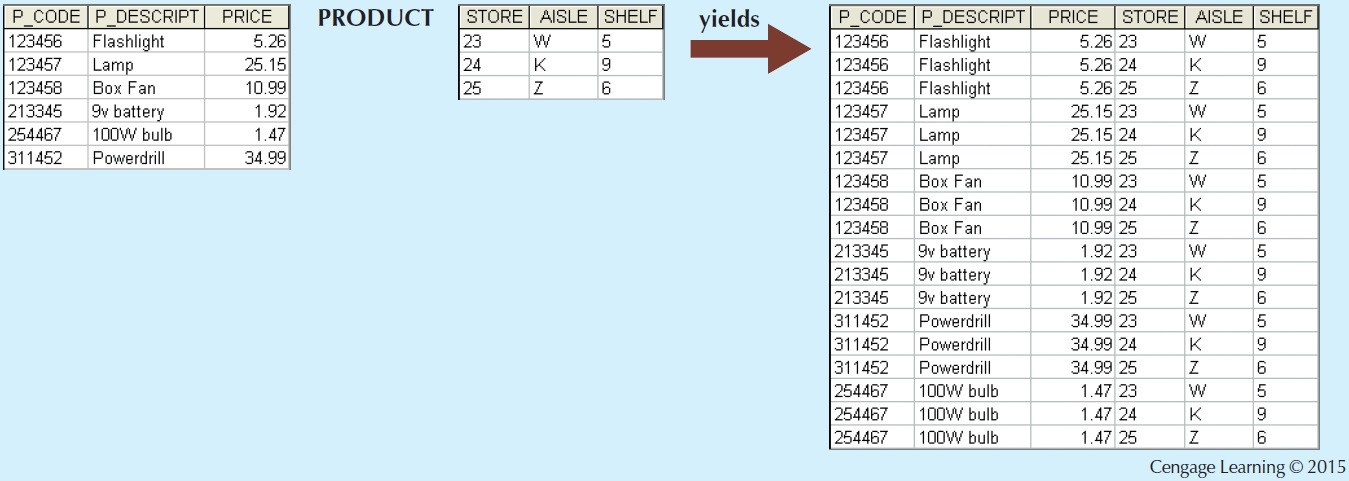


Figure 3.10 - Two Tables That Will Be

Used in JOIN Illustrations

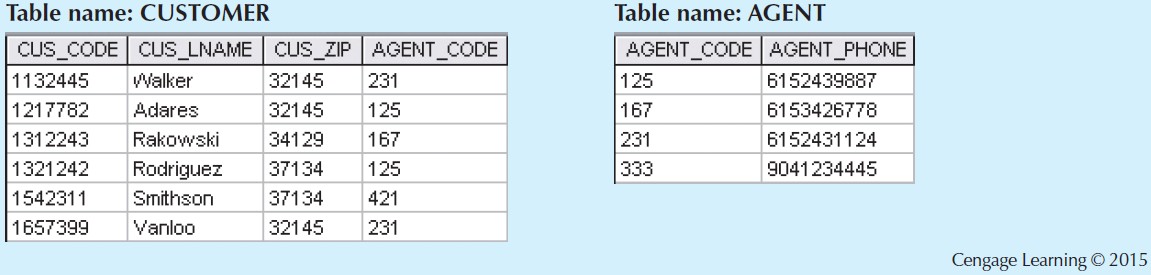
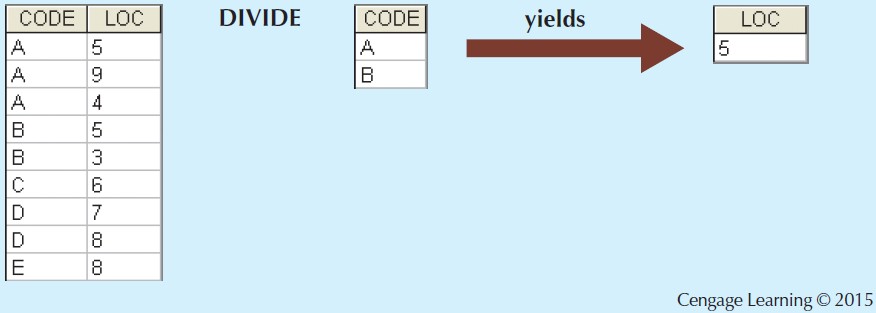


Figure 3.16 - Divide



Data Dictionary and the System Catalog

**Data dictionary**: Description of all tables in the database created by the user and designer

**System catalog**: System data dictionary that describes all objects within the database

Homonyms and synonyms must be avoided to lessen confusion

**Homonym**: Same name is used to label different attributes

**Synonym**: Different names are used to describe the same attribute

Relationships within the Relational

Database

1:M relationship - Norm for relational databases

1:1 relationship - One entity can be related to only one other entity and vice versa

Many-to-many (M:N) relationship - Implemented by creating a new entity in 1:M relationships with the original entities

**Composite entity** (**Bridge** or **associative entity**): Helps avoid problems inherent to M:N relationships, includes the primary keys of tables to be linked

Figure 3.21 - The 1:1 Relationship between

PROFESSOR and DEPARTMENT

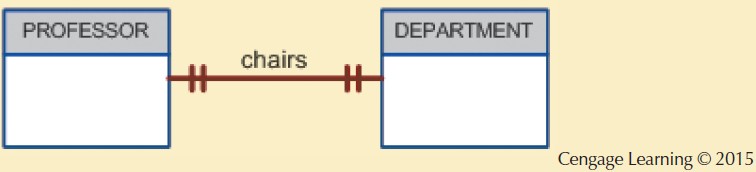


Figure 3.26 - Changing the M:N Relationship to Two 1:M Relationships

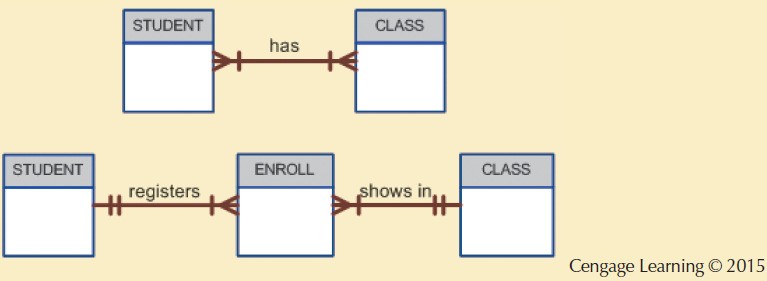
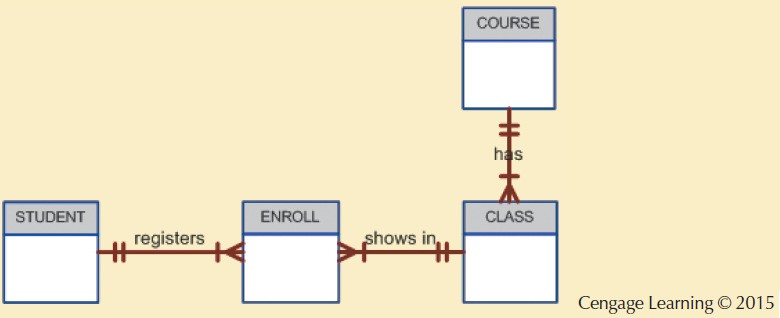


Figure 3.27 - The Expanded ER Model



Data Redundancy

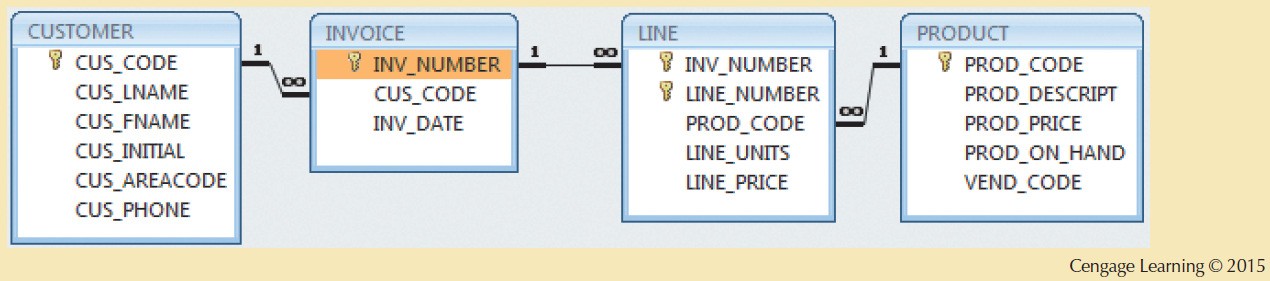
Relational database facilitates control of data redundancies through use of foreign keys

To be controlled except the following circumstances

Data redundancy must be increased to make the database serve crucial information purposes

Exists to preserve the historical accuracy of the data

Figure 3.30 - The Relational Diagram for the Invoicing System



Index

Orderly arrangement to logically access rows in a table

**Index key**: Index’s reference point that leads to data location identified by the key

**Unique index**: Index key can have only one pointer value associated with it

Each index is associated with only one table