WelTec
Te Whare Wānanga o te Awakairangi

西安科技大学高新学院
XI'AN KEDAGAOXIN UNIVERSITY

IT5507 Fundamentals of Data Science

Chapter 3 The Relational Database Model

CENGAGE

Learning Objectives

- After completing this chapter, you will be able to:
  - Describe the relational database model's logical structure
  - Identify the relational model's basic components and explain the structure, contents, and characteristics of a relational table
  - Use relational database operators to manipulate relational table contents
  - Explain the purpose and components of the data dictionary and system catalog
  - Identify appropriate entities and then the relationships among the entities in the relational database model
  - Describe how data redundancy is handled in the relational database model
  - Explain the purpose of indexing in a relational database

A Logical View of Data

- **Logical Data Representation:** Relational Database Model enables a logical portrayal of data and its relationships.
- **Design Simplicity:** Logical simplicity leads to straightforward and effective methodologies for designing databases.
- **Data Relationships:** The creation of relationships, based on a logical construct known as a relation, facilitates a cohesive logical view of the data.
- **Strategic Insights:** A logical view of data is pivotal for gaining strategic insights, ensuring efficient decision-making and enhancing the overall understanding of complex information structures.

Tables and Their Characteristics

Table 3.1	Characteristics of a Relational Table
1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (tuple) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each intersection of a row and column represents a single data value.
5	All values in a column must conform to the same data format.
6	Each column has a specific range of values known as the attribute domain.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or combination of attributes that uniquely identifies each row.

CENGAGE

Keys

- **Key Principles:** Keys, like a social security number in a citizens' database, consist of one or more attributes, dictating and uniquely identifying rows, fostering relationships among tables and upholding data integrity.
- **Primary Key Significance:** In a student information database, the combination of Student ID and Course ID serves as the primary key, uniquely identifying each enrollment record and ensuring data integrity within the relational structure.

- **Key Principles:** In a library database, a book's ISBN (International Standard Book Number) can serve as a key, ensuring each book is uniquely identifiable and facilitating relationships with information about borrowers or authors.
- **Primary Key Significance:** The ISBN, in this case, acts as the primary key, uniquely identifying each book entry and maintaining the integrity of the library database.

Dependencies

- **Determination in Action:** In an employee database, the Social Security Number (SSN) can be a determinant, where knowing it allows determination of the employee's full name, establishing a functional dependence crucial for relational integrity.

- **Example of Functional Dependence:** If the SSN is the determinant, the employee's full name becomes the dependent, showcasing a functional dependence where one attribute's value determines another’s.

- **Full Functional Dependence Illustration:** In this scenario, the entire collection of attributes in the determinant (SSN) is necessary for establishing the relationship and determining the dependent attribute (full name).

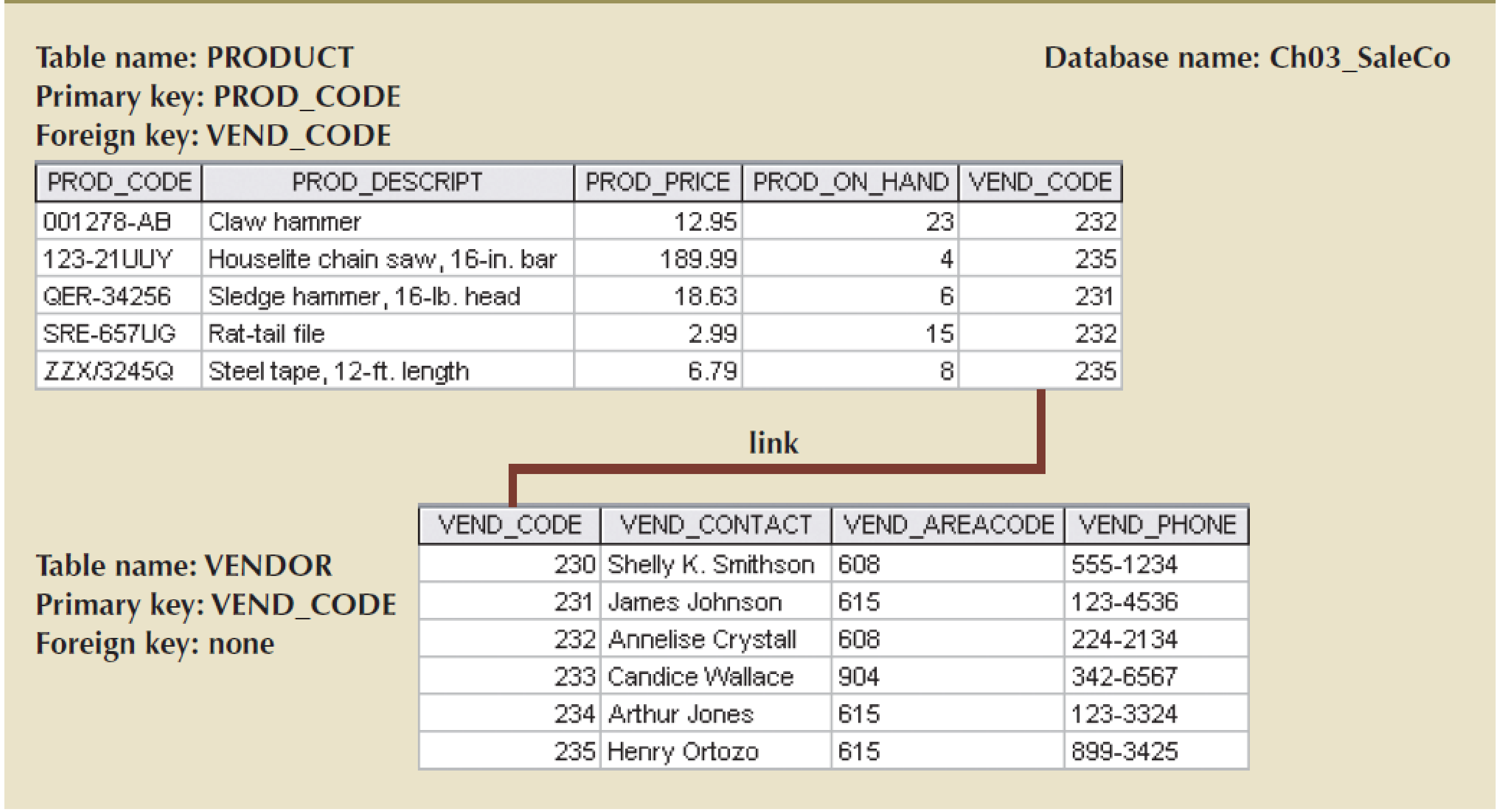
Types of Keys (1 of 3)

- **Composite Key Example:** In a university database, a composite key could be formed by combining the Student ID and Course Code, ensuring each enrollment record is uniquely identified.
- **Superkey Illustration:** A superkey might include the combination of Student ID, Course Code, and Semester, uniquely identifying any enrollment record in the table.
- **Candidate Key in Action:** Among superkeys, a candidate key would be a minimal combination like Student ID and Course Code, providing unique identification without unnecessary attributes.
- **Entity Integrity Exemplified:** In a student table, entity integrity ensures each student has a unique Student ID, and all values in the primary key (Student ID) are distinct.
- **Referential Integrity Scenario:** In a department table, referential integrity ensures that every reference to a Department ID in a course table is valid, creating a relationship between entities.
- **Foreign Key Usage:** If the Department ID from the department table is placed into the course table, it becomes a foreign key establishing a connection between the two tables.
- **Secondary Key Purpose:** A secondary key, like indexing courses based on their names, is used strictly for data retrieval without the uniqueness constraints of a primary key.

Think of a library catalog where books are indexed by genre (e.g., "Mystery" or "Science Fiction"). This genre index serves as a secondary key, allowing readers to retrieve books based on categories without requiring each genre to be unique like a primary key.

Types of Keys (2 of 3)

FIGURE 3.2 AN EXAMPLE OF A SIMPLE RELATIONAL DATABASE



Database name: Ch03_SaleCo

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: VEND_CODE

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	18.63	6	231
SRE-657UG	Rat-tail file	2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	6.79	8	235

Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

Types of Keys (3 of 3)

Table 3.3	Relational Database Keys
Key Type	**Definition**
Superkey	An attribute or combination of attributes that uniquely identifies each row in a table.
Candidate key	A minimal (irreducible) superkey; a superkey that does not contain a subset of attributes that is itself a superkey.
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row; cannot contain null entries.
Foreign key	An attribute or combination of attributes in one table whose values must either match the primary key in another table or be null.
Secondary key	An attribute or combination of attributes used strictly for data retrieval purposes.

Integrity Rules (1 of 4)

- Relational database integrity rules are very important to good database design
  - Relational database management systems (RDBMSs) enforce integrity rules automatically
    - Much safer to make sure the application design conforms to entity and referential integrity rules
    - Importance of Integrity Rules: Ensuring the reliability of a relational database, integrity rules play a vital role in maintaining data accuracy. While RDBMSs automatically enforce these rules, it's crucial for application design to align with entity and referential integrity for a safer and more dependable database.

Ensuring Accuracy with Integrity Rules: Consider a university database where the student table has a primary key for student IDs. Enforcing entity integrity ensures that each student is uniquely identified. Now, if another table references student IDs, referential integrity ensures that any reference corresponds to a valid student ID in the student table, preventing data inconsistencies. These rules, when followed, contribute to a reliable and accurate database.

Integrity Rules (2 of 4)

Table 3.4	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 can it be null; in short, all invoices are uniquely identified by their invoice number.
Referential Integrity	**Description**
Requirement	A foreign key may have either a null entry, as long as it is not a part of its table's primary key, or an entry that matches the primary key value in a table to which it is related (every non-null foreign key value must reference an existing primary key value).
Purpose	It is possible for an attribute not to have a corresponding value, but it will be impossible to have an invalid entry; the enforcement of the referential integrity rule makes it impossible to delete a row in one table whose primary key has mandatory matching foreign key values in another table.
Example	A customer might not yet have an assigned sales representative (number), but it will be impossible to have an invalid sales representative (number).

CENGAGE

Integrity Rules (3 of 4)

FIGURE 3.3 AN ILLUSTRATION OF INTEGRITY RULES

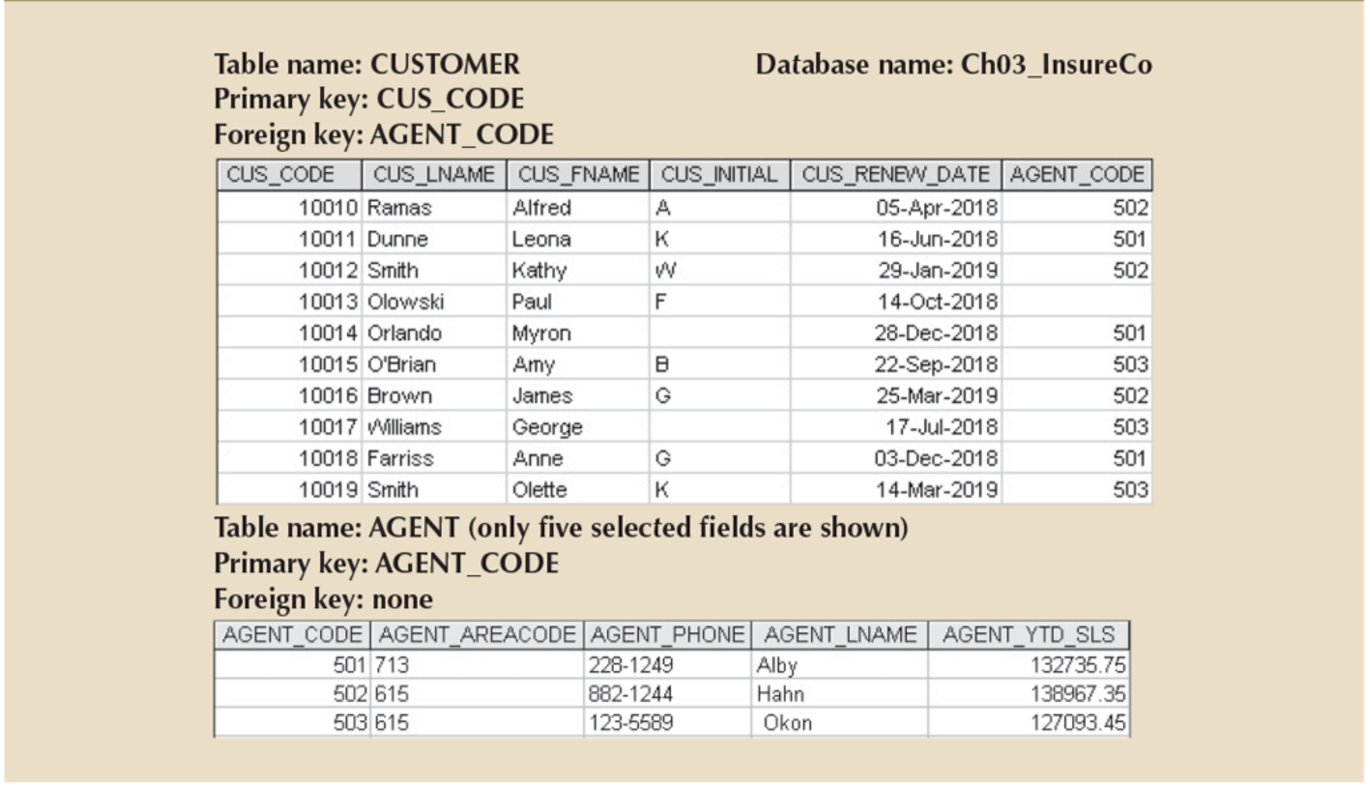


Table name: CUSTOMER **Database name: Ch03_InsureCo**
Primary key: CUS_CODE
Foreign key: AGENT_CODE

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	05-Apr-2018	502
10011	Dunne	Leona	K	16-Jun-2018	501
10012	Smith	Kathy	W	29-Jan-2019	502
10013	Olowski	Paul	F	14-Oct-2018	
10014	Orlando	Myron		28-Dec-2018	501
10015	O'Brian	Amy	B	22-Sep-2018	503
10016	Brown	James	G	25-Mar-2019	502
10017	Williams	George		17-Jul-2018	503
10018	Farriss	Anne	G	03-Dec-2018	501
10019	Smith	Olette	K	14-Mar-2019	503

Table name: AGENT (only five selected fields are shown)
Primary key: AGENT_CODE
Foreign key: none

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SLS
501	713	228-1249	Alby	132735.75
502	615	882-1244	Hahn	138967.35
503	615	123-5589	Okon	127093.45

CENGAGE

Integrity Rules (4 of 4)

- Ways to handle nulls
  - Flags
    - Special codes used to indicate the absence of some value
  - Constraints
    - 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

Example of Using Flags: In a database tracking employee attendance, a special code like "-1" could be assigned to signify that an employee did not clock in on a particular day.
Example of Using Constraints: Applying a NOT NULL constraint to the "Email" column in a user database ensures that each user record must have a valid email address, preventing the absence of this crucial information. Similarly, a UNIQUE constraint on the "Username" column guarantees that no two users share the same username, maintaining data consistency.

Relational Algebra

- Theoretical way of manipulating table contents using relational operators
  - Relvar: variable that holds a relation
    - Heading contains the names of the attributes
    - Body contains the relation
  - Relational operators have the property of closure
    - Closure: use of relational algebra operators on existing relations produces new relations

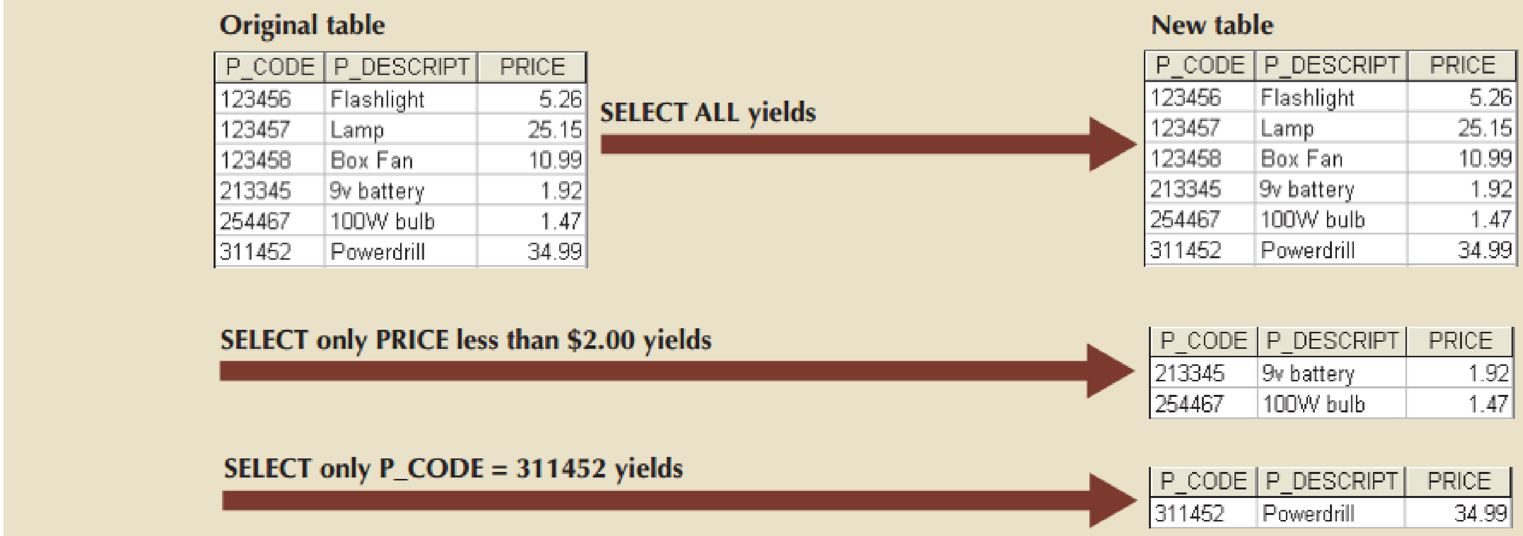
In the world of databases:
•**Relvar** is like a storage box that holds information (a table).
•**Heading** is the list of attributes (column names) in the box.
•**Body** is the actual data stored in the box.
Now, think of relational algebra as a set of actions you can perform on these boxes:
•**Operators** are the actions you take, like selecting specific attributes or combining two boxes.
•**Closure** means that when you perform these actions, you get a new box with its own heading and body.
So, it's like playing with boxes of information – you can choose what to look at, combine them, and create new boxes based on these operations.

Relational Set Operators (1 of 11)

- **Select (restrict):** Think of it as putting a filter to get only specific rows from a table. For example, you can select all customers who are from New York.
- **Project:** This is like trimming down your table to just the columns you're interested in. For instance, if you have a table with customer information but only want to see their names and emails, you'd project those two columns.
- **Union:** Imagine stacking two tables on top of each other, but making sure you don't have any duplicate rows. So if you have one table with customer names and another with employee names, the union would combine them into a single table with all unique names.
- **Intersect:** This operation gives you only the rows that are common between two tables. For example, if you have a table of products that are both in stock and on sale, and another table of products that are on sale, intersecting them would give you only the products that are both in stock and on sale.

Relational Set Operators (2 of 11)

FIGURE 3.4 SELECT


SELECT ALL yields
SELECT only PRICE less than $2.00 yields
SELECT only P_CODE = 311452 yields

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

New table

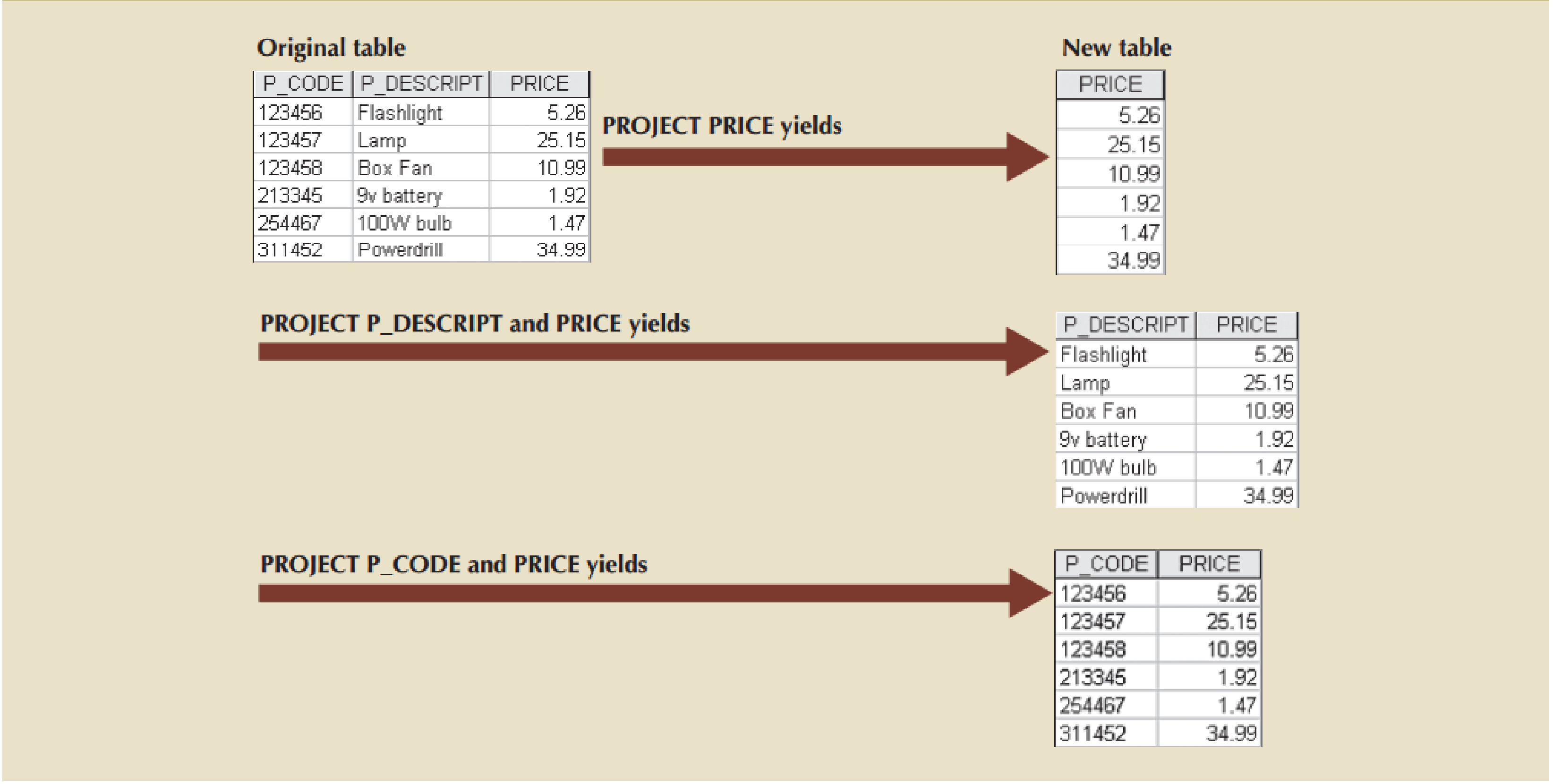
P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

P_CODE	P_DESCRIPT	PRICE
213345	9v battery	1.92
254467	100W bulb	1.47

P_CODE	P_DESCRIPT	PRICE
311452	Powerdrill	34.99

Relational Set Operators (3 of 11)

FIGURE 3.5 PROJECT



Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

PROJECT PRICE yields

New table

PRICE
5.26
25.15
10.99
1.92
1.47
34.99

PROJECT P_DESCRIPT and PRICE yields

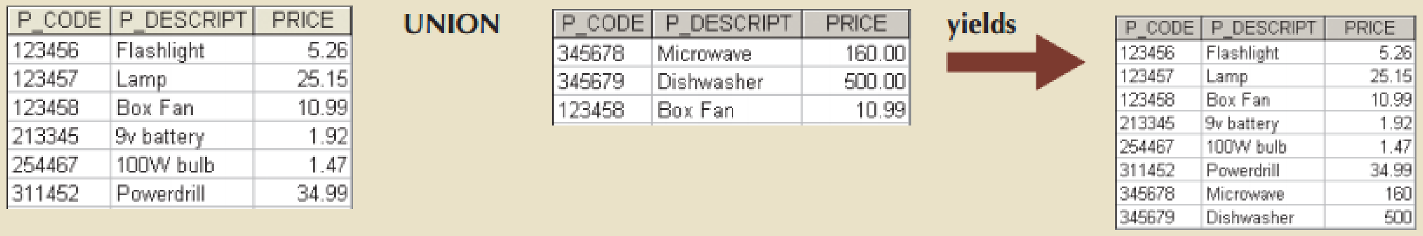
P_DESCRIPT	PRICE
Flashlight	5.26
Lamp	25.15
Box Fan	10.99
9v battery	1.92
100W bulb	1.47
Powerdrill	34.99

PROJECT P_CODE and PRICE yields

P_CODE	PRICE
123456	5.26
123457	25.15
123458	10.99
213345	1.92
254467	1.47
311452	34.99

Relational Set Operators (4 of 11)

FIGURE 3.6 UNION



P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

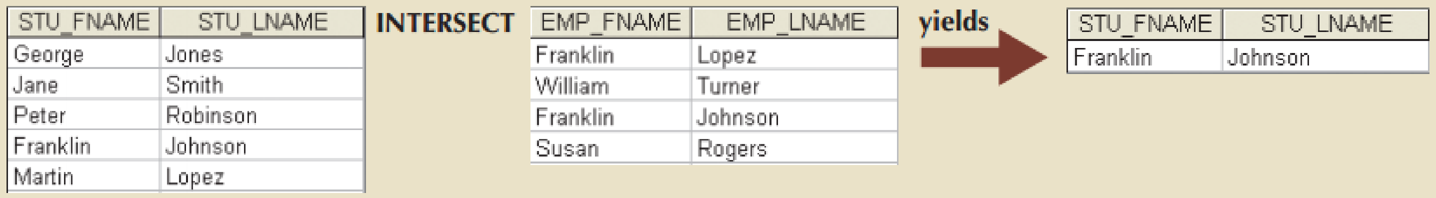
UNION

P_CODE	P_DESCRIPT	PRICE
345678	Microwave	160.00
345679	Dishwasher	500.00
123458	Box Fan	10.99

yields

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99
345678	Microwave	160
345679	Dishwasher	500

FIGURE 3.7 INTERSECT



STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

INTERSECT

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

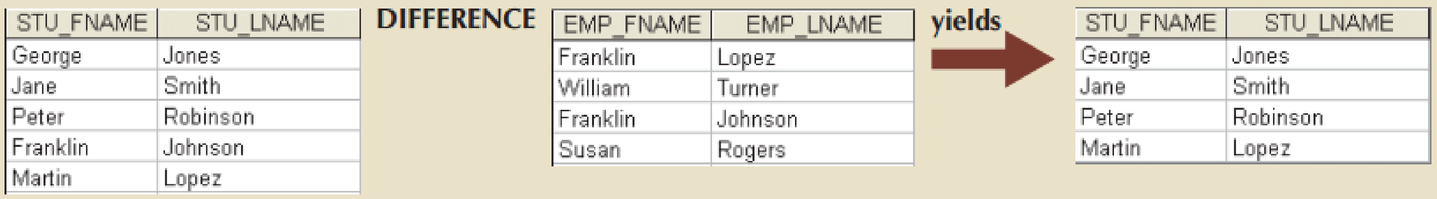
STU_FNAME	STU_LNAME
Franklin	Johnson

Relational Set Operators (5 of 11)

- **Difference (Except):** This operation gives you the rows that exist in one table but not in another. For example, if you have a table of all products and another table of products that are out of stock, the difference would give you only the products that are still in stock.
- **Product/ Cartesian Product (Cross Join):** This operation combines every row from one table with every row from another table, resulting in a new table. For instance, if you have a table of colors and a table of sizes, the Cartesian product would create a table with all possible color and size combinations.

Relational Set Operators (6 of 11)

FIGURE 3.8 DIFFERENCE



STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

DIFFERENCE

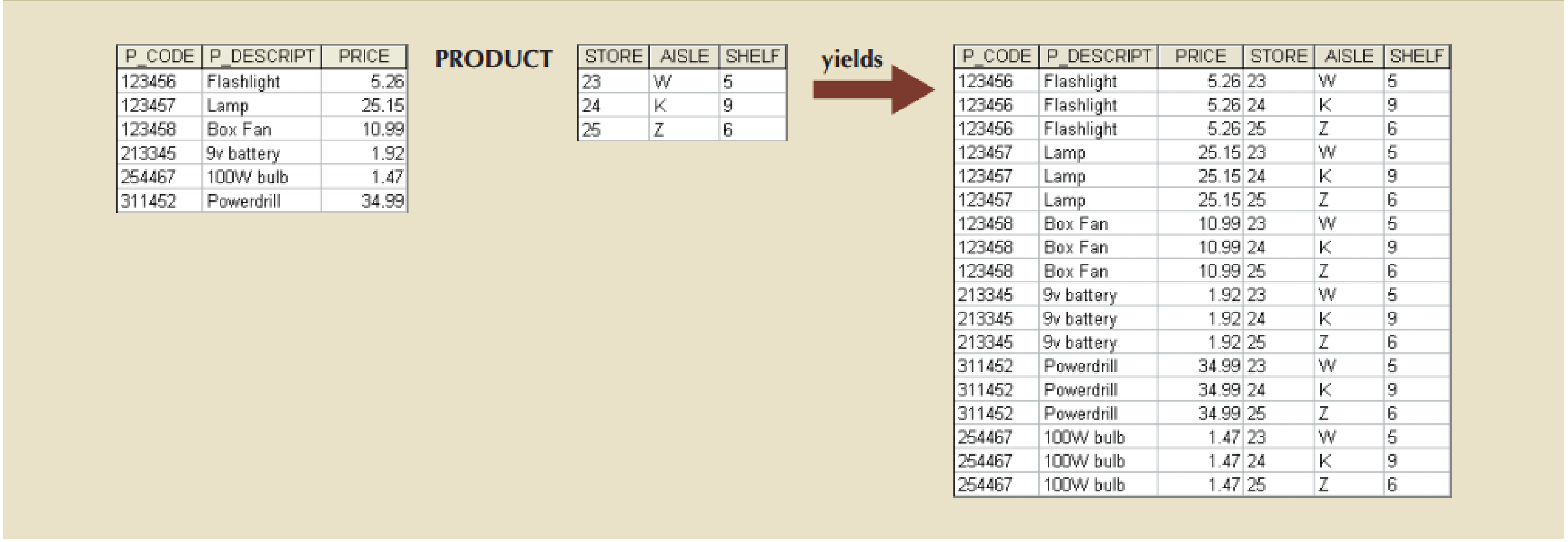
EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Martin	Lopez

Relational Set Operators (7 of 11)

FIGURE 3.9 PRODUCT



P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

PRODUCT

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

yields

P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	5.26	23	W	5
123456	Flashlight	5.26	24	K	9
123456	Flashlight	5.26	25	Z	6
123457	Lamp	25.15	23	W	5
123457	Lamp	25.15	24	K	9
123457	Lamp	25.15	25	Z	6
123458	Box Fan	10.99	23	W	5
123458	Box Fan	10.99	24	K	9
123458	Box Fan	10.99	25	Z	6
213345	9v battery	1.92	23	W	5
213345	9v battery	1.92	24	K	9
213345	9v battery	1.92	25	Z	6
311452	Powerdrill	34.99	23	W	5
311452	Powerdrill	34.99	24	K	9
311452	Powerdrill	34.99	25	Z	6
254467	100W bulb	1.47	23	W	5
254467	100W bulb	1.47	24	K	9
254467	100W bulb	1.47	25	Z	6

Relational Set Operators (8 of 11)

Joins are operations in relational databases that intelligently combine information from two or more tables. Different types of joins include:

1. **Natural Join**: Links tables by selecting only the rows with common values in their common attribute.
2. **Equijoin**: Links tables based on an equality condition that compares specified columns of each table.
3. **Theta Join**: Links tables using an inequality comparison operator.
4. **Inner Join**: Only returns matched records from the tables that are being joined.
5. **Outer Join**: Retains matched pairs and leaves unmatched values in the other table as null.
    1. **Left Outer Join**: Yields all rows in the first table, including those without a matching value in the second table.
    2. **Right Outer Join**: Yields all rows in the second table, including those without matching values in the first table.

Relational Set Operators (9 of 11)

- Divide
  - Uses one double-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
  - This operation is a bit more complex. It is used to find records in one table for which there are matching records in another table for every record in the first table. An example could be finding customers who have purchased all available products.

Relational Set Operators (10 of 11)

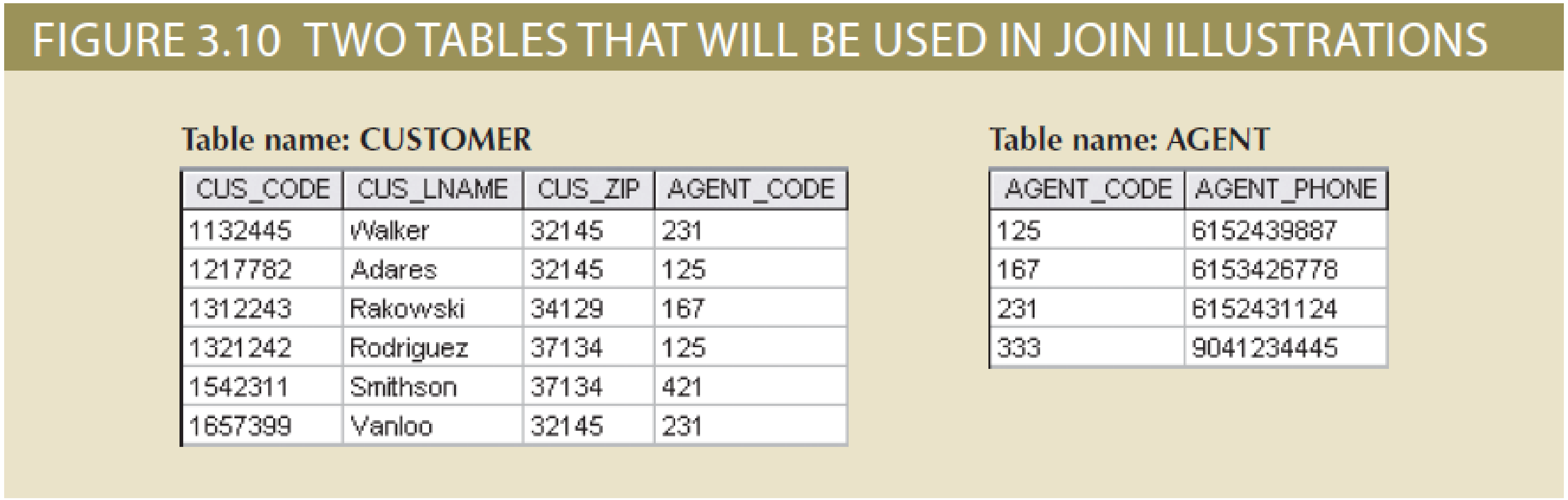


FIGURE 3.10 TWO TABLES THAT WILL BE USED IN JOIN ILLUSTRATIONS

Table name: CUSTOMER

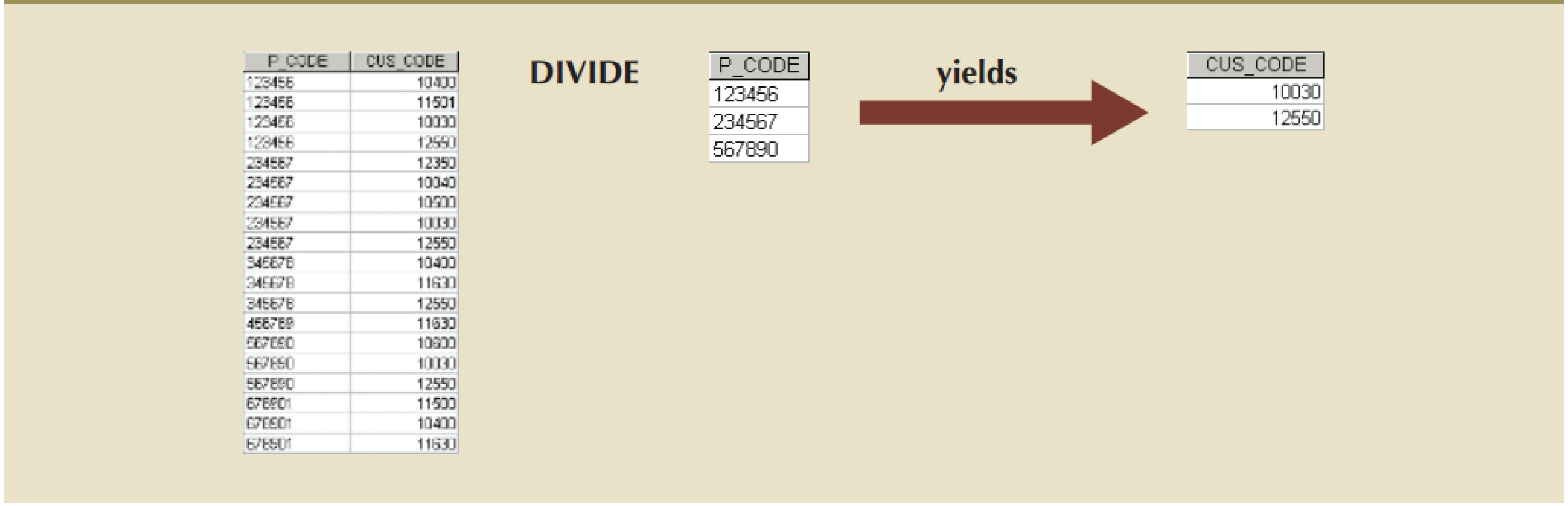
CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Relational Set Operators (11 of 11)

FIGURE 3.16 DIVIDE


DIVIDE
yields

P_CODE	CUS_CODE
123456	10400
123456	11501
123456	10030
123456	12550
234567	12350
234567	10040
234567	10500
234567	10030
234567	12550
345678	10400
345678	11630
345678	12550
456789	11630
567890	10900
567890	10030
567890	12550
678901	11500
678901	10400
678901	11630

DIVIDE

P_CODE
123456
234567
567890

yields

CUS_CODE
10030
12550

Data Dictionary and the System Catalog

A data dictionary is a comprehensive description of all tables in a database, providing details created by both the user and designer. The system catalog, part of the data dictionary, encompasses information about all objects within the database. To prevent confusion, it is crucial to avoid homonyms (same name for different attributes) and synonyms (different names for the same attribute). This meticulous documentation ensures clarity and consistency in database management.

- 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 of 4)

- **One-to-Many (1:M):**

Definition: A norm for relational databases where one entity is associated with many instances of another entity.

Example: A university database where one department has many professors.

- **One-to-One (1:1):**

Definition: Each entity can be related to only one other entity, and vice versa.

Example: A database for employee records where each employee has only one office.

- **Many-to-Many (M:N):**

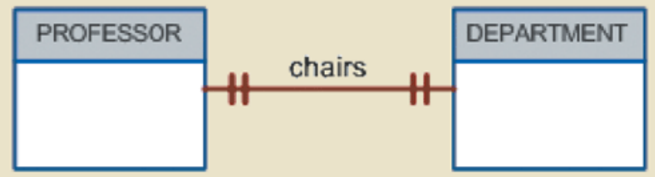
Definition: Implemented by creating a new entity in 1:M relationships with the original entities.

Example: A database for courses and students where one student can enroll in multiple courses, and a course can have multiple students. A composite entity (bridge table) might include the primary keys of both the "Student" and "Course" tables.

CENGAGE

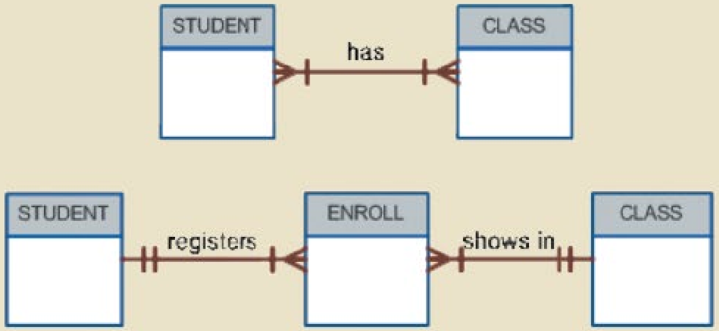
Relationships within the Relational Database (2 of 4)

FIGURE 3.21 THE 1:1 RELATIONSHIP BETWEEN PROFESSOR AND DEPARTMENT



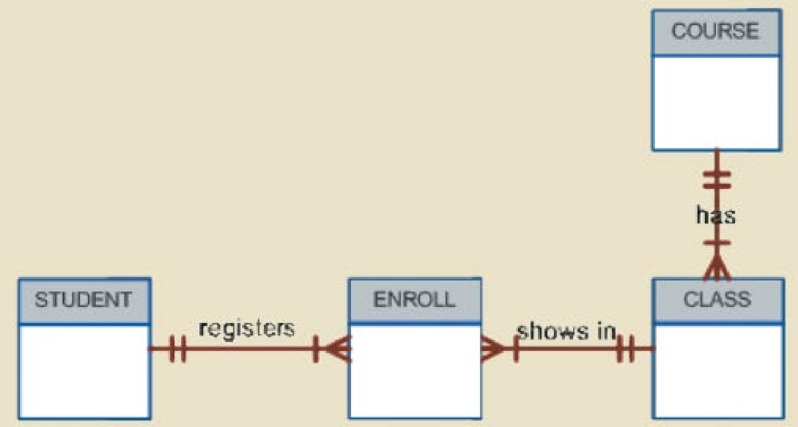
Relationships within the Relational Database (3 of 4)

FIGURE 3.26 CHANGING THE M:N RELATIONSHIPS TO TWO 1:M RELATIONSHIPS



Relationships within the Relational Database (4 of 4)

FIGURE 3.27 THE EXPANDED ER MODEL



Data Redundancy Revisited (1 of 2)

- **Control of Data Redundancies in Relational Databases:**

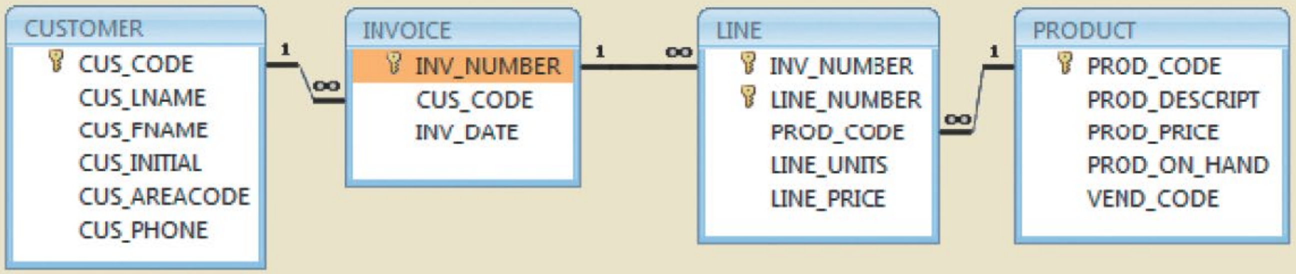
- **Foreign Keys Example:** Consider two tables, 'Customers' and 'Orders.' The 'CustomerID' in the 'Orders' table serves as a foreign key linked to the 'CustomerID' in the 'Customers' table. This prevents duplicating customer information in each order.

- **Common Attributes Example:** If 'ProductID' is a common attribute between 'Products' and 'Sales,' linking these tables through this attribute helps control data redundancies, ensuring consistency in product-related information.

- Exceptions to Controlling Data Redundancy:
- **Information Purposes Example:** In a reporting database, redundant information might be stored to facilitate quicker data retrieval and reporting, emphasizing performance over normalization.
- **Historical Accuracy Example:** In a historical database, redundant data might be stored to preserve the state of information at different points in time, ensuring accurate historical reporting.

Data Redundancy Revisited (2 of 2)

FIGURE 3.30 THE RELATIONAL DIAGRAM FOR THE INVOICING SYSTEM



Indexes

- **Database Indexing:**
- **Index Key:** A database index provides an orderly arrangement to logically access rows in a table. The index key is the reference point that leads to the data location identified by the key.
- **Unique Index Example:** In a 'Students' table, the 'StudentID' can serve as a unique index key. Each student ID uniquely identifies a student, and the index ensures efficient retrieval of student data based on this key.
- **Efficient Retrieval:** Indexing ensures efficient retrieval of data by providing an orderly arrangement, reducing the need for full table scans.
- **Faster Queries:** By creating an index on frequently queried columns, database searches and operations become faster.
- **Enhanced Performance:** Indexing enhances database performance, particularly in large datasets, by optimizing data access pathways.

Codd’s Relational Database Rules (1 of 2)

Codd's rules, fundamental to relational databases, emphasize criteria such as information rule, guaranteed access, systematic treatment of null values, comprehensive data sublanguage, and integrity constraints, ensuring a standardized and reliable approach to database management.

Table 13.8	**Dr. Codd’s 12 Relational Database Rules**	
Rule	Rule Name	Description
1	Information	All information in a relational database must be logically represented as column values in rows within tables.
2	Guaranteed access	Every value in a table is guaranteed to be accessible through a combination of table name, primary key value, and column name.
3	Systematic treatment of nulls	Nulls must be represented and treated in a systematic way, independent of data type.
4	Dynamic online catalog based on the relational model	The metadata must be stored and managed as ordinary data—that is, in tables within the database; such data must be available to authorized users using the standard database relational language.
5	Comprehensive data sublanguage	The relational database may support many languages; however, it must support one well-defined, declarative language as well as data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization, and transaction management (begin, commit, and rollback).
6	View updating	Any view that is theoretically updatable must be updatable through the system.
7	High-level insert, update, and delete	The database must support set-level inserts, updates, and deletes.

CENGAGE

Codd’s Relational Database Rules (2 of 2)

Table 13.8	Dr. Codd’s 12 Relational Database Rules	
Rule	Rule Name	Description
8	Physical data independence	Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9	Logical data independence	Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of columns or inserting columns).
10	Integrity independence	All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11	Distribution independence	The end users and application programs are unaware of and unaffected by the data location (distributed vs. local databases).
12	Nonsubversion	If the system supports low-level access to the data, users must not be allowed to bypass the integrity rules of the database.
13	Rule zero	All preceding rules are based on the notion that to be considered relational, a database must use its relational facilities exclusively for management.

CENGAGE

Summary

- Tables are the basic building blocks of a relational database
  - Keys are central to the use of relational tables
    - Each table row must have a primary key
  - Although tables are independent, they can be linked by common attributes
- The relational model supports several relational algebra functions
  - A relational database performs much of the data manipulation work behind the scenes
  - Once you know the basics of relational databases, you can concentrate on design