

Chapter 3

The Relational Database Model

Learning Objectives (1 of 2)

- In this chapter, you 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 (2 of 2)

- In this chapter, you 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

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.

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
- 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 (1 of 2)

- **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 (2 of 2)

- **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
- **Secondary key:** Key used strictly for data retrieval purposes

Figure 3.2 - An Example of a Simple Relational Database

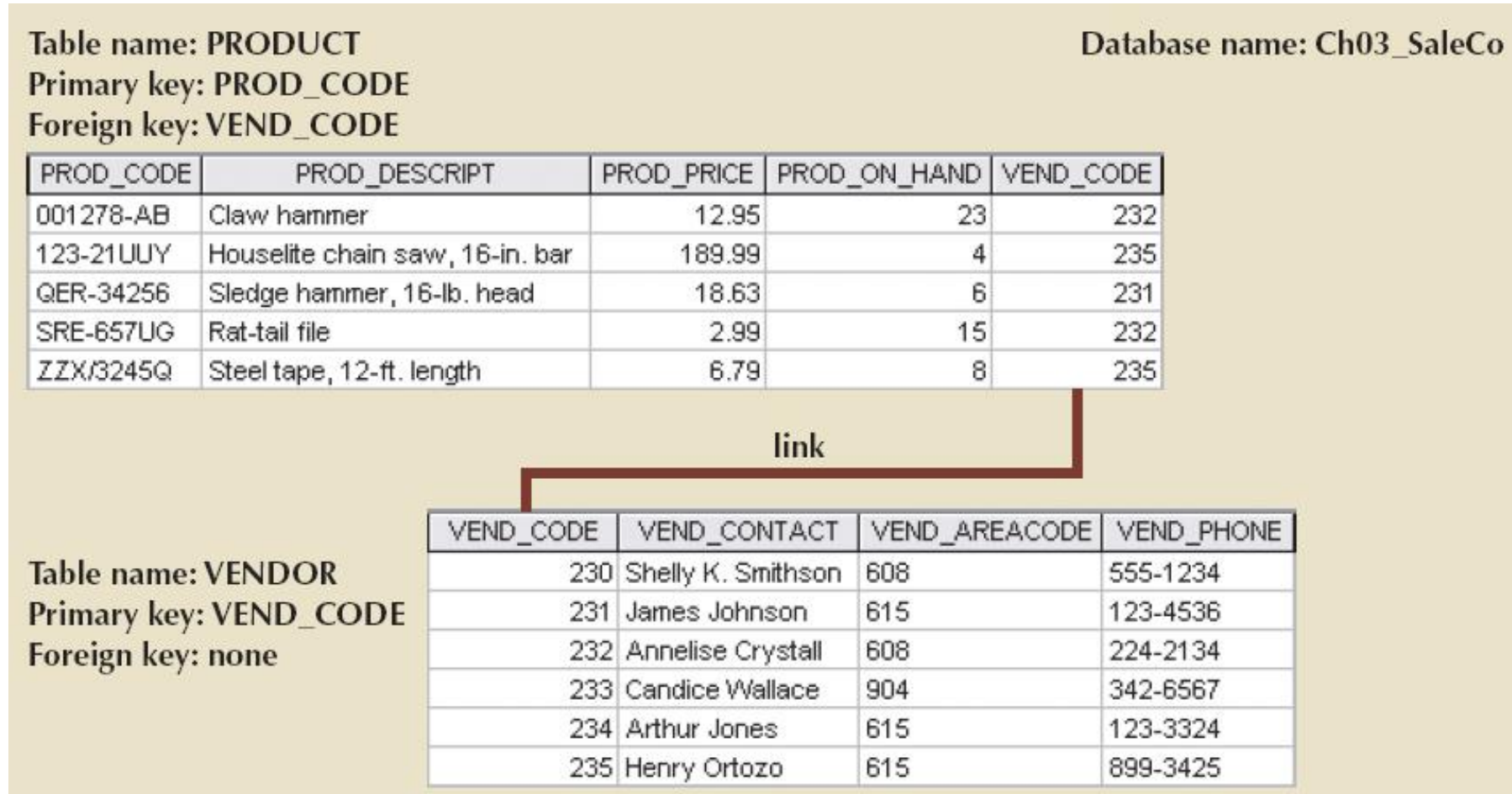


Table 3.3 - Relational Database Keys

KEY TYPE	DEFINITION
Super key	An attribute or combination of attributes that uniquely identifies each row in a table
Candidate key	A minimal (irreducible) super key; a super key that does not contain a subset of attributes that is itself a super key
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

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 <i>must</i> reference an <i>existing primary</i> 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).

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-2016	502
10011	Dunne	Leona	K	16-Jun-2016	501
10012	Smith	Kathy	W	29-Jan-2017	502
10013	Olowski	Paul	F	14-Oct-2016	
10014	Orlando	Myron		28-Dec-2016	501
10015	O'Brian	Amy	B	22-Sep-2016	503
10016	Brown	James	G	25-Mar-2017	502
10017	Williams	George		17-Jul-2016	503
10018	Farriss	Anne	G	03-Dec-2016	501
10019	Smith	Olette	K	14-Mar-2017	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

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

Select (Restrict)

- Unary operator that yields a horizontal subset of a table

Project

- Unary operator that yields a vertical subset of a table

Union

- 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

Intersect

- Yields only the rows that appear in both tables
- Tables must be union-compatible to yield valid results

Figure 3.4 - Select

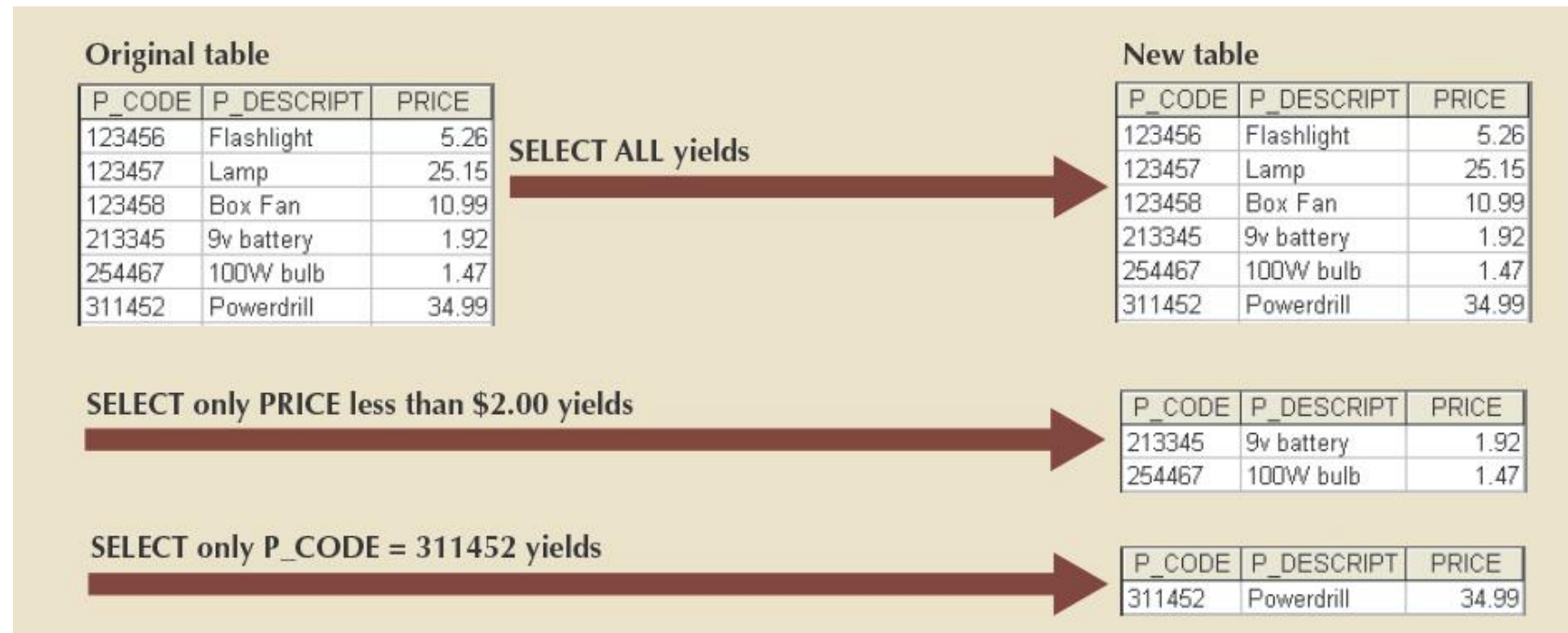


Figure 3.5 - Project

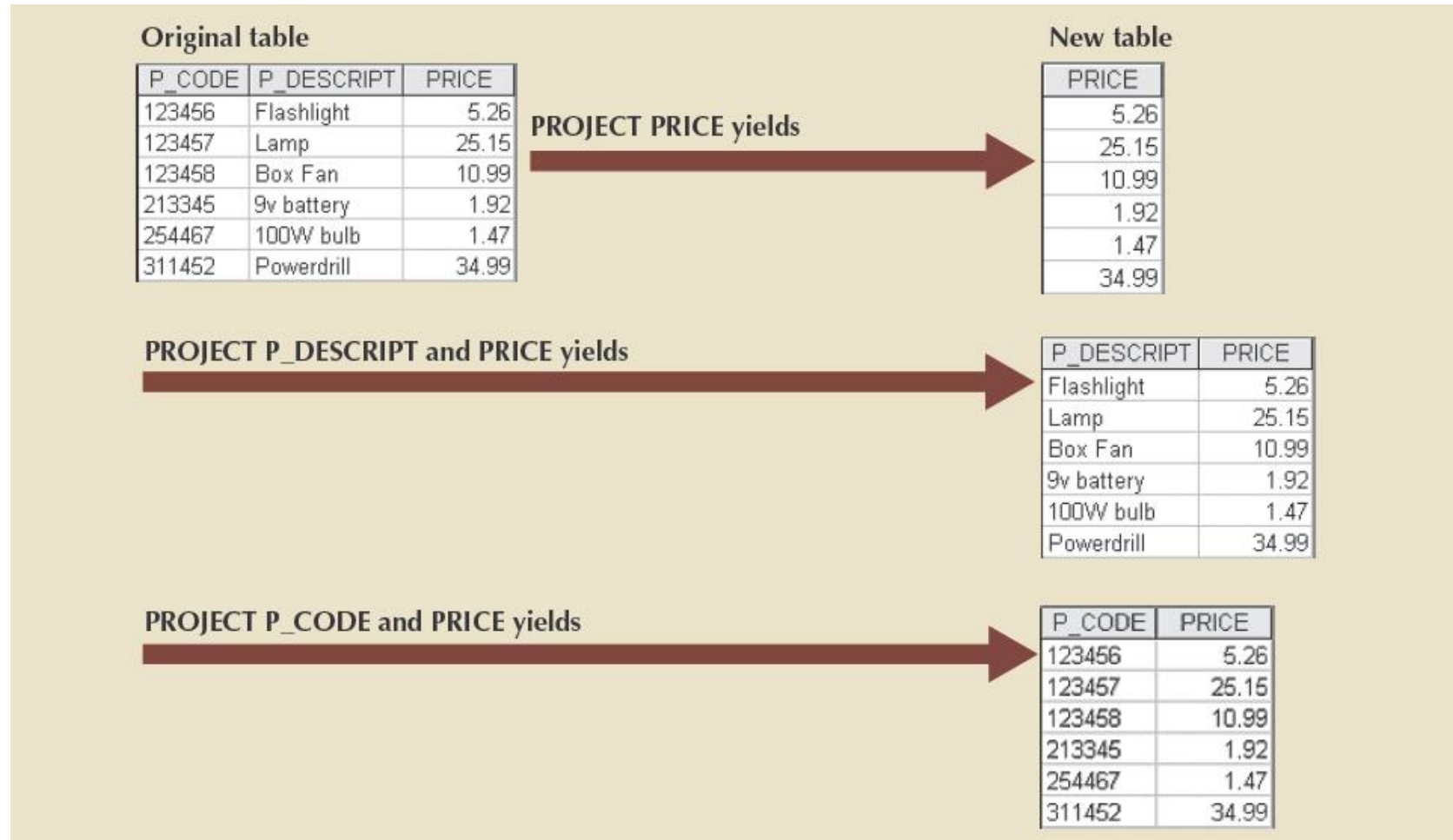


Figure 3.6 - Union and Figure 3.7 - Intersect

Figure 3.6 - Union



P_CODE	P_DESCRIPT	PRICE	UNION			yields	P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26					123456	Flashlight	5.26
123457	Lamp	25.15					123457	Lamp	25.15
123458	Box Fan	10.99					123458	Box Fan	10.99
213345	9v battery	1.92					213345	9v battery	1.92
254467	100W bulb	1.47					254467	100W bulb	1.47
311452	Powerdrill	34.99					311452	Powerdrill	34.99
							345678	Microwave	160
							345679	Dishwasher	500.00
							123458	Box Fan	10.99

Figure 3.7 - Intersect

STU_FNAME	STU_LNAME	INTERSECT	EMP_FNAME	EMP_LNAME	yields 	STU_FNAME	STU_LNAME
George	Jones		Franklin	Lopez		Franklin	Johnson
Jane	Smith		William	Turner			
Peter	Robinson		Franklin	Johnson			
Franklin	Johnson		Susan	Rogers			
Martin	Lopez						

Relational Set Operators (1 of 2)

- **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

Figure 3.8 – Difference

Figure 3.8 – Difference

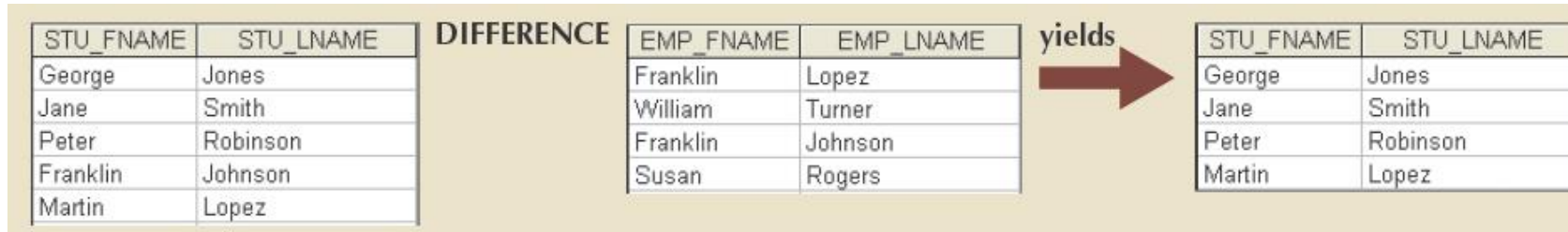


Figure 3.9 - Product

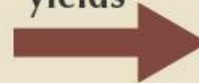
Figure 3.9 - Product

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 (2 of 2)

- **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 (1 of 2)

- **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 (2 of 2)

- **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 3.10 - Two Tables That Will Be Used in JOIN Illustrations

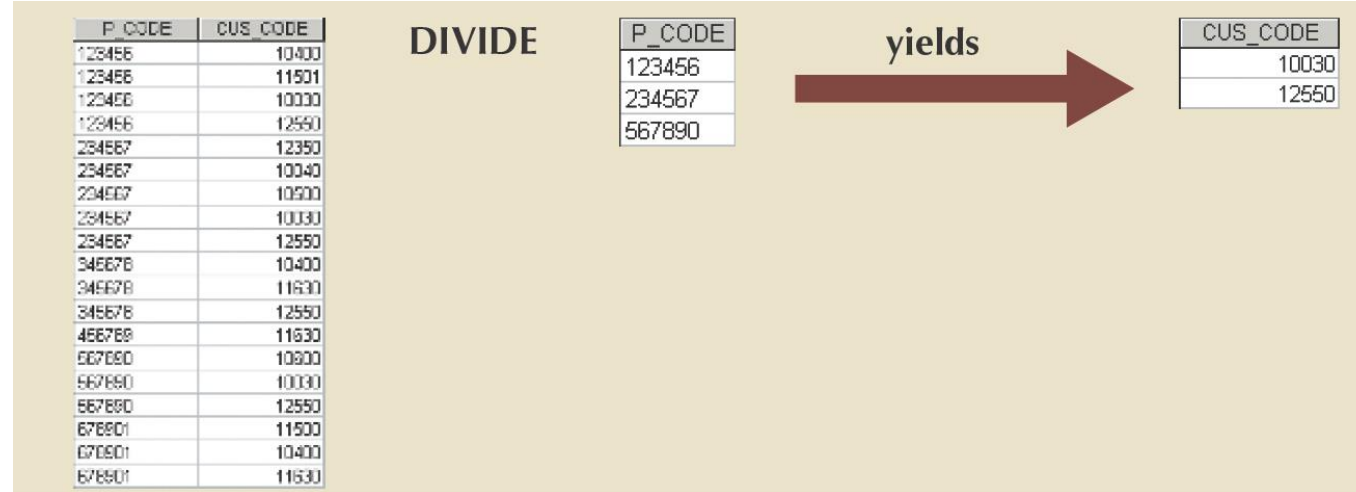
Table name: CUSTOMER

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

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

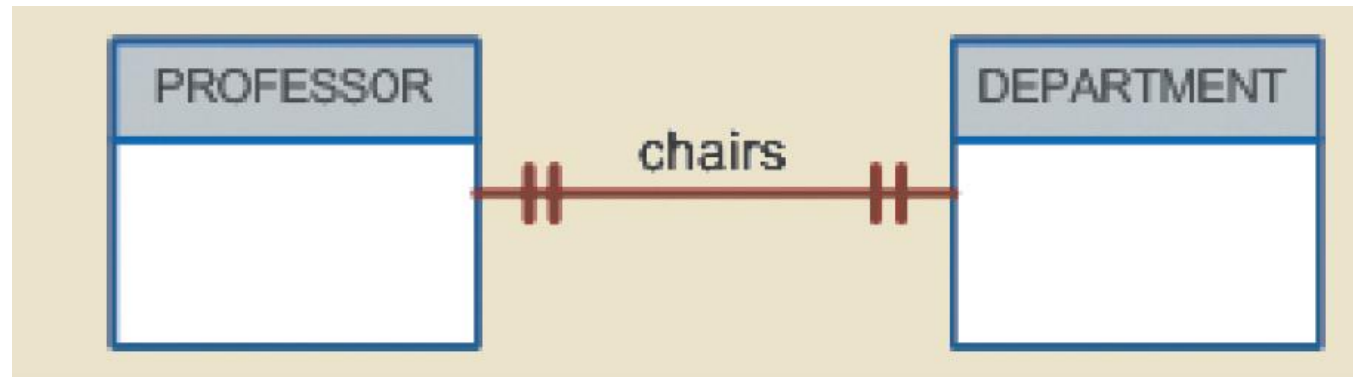


Figure 3.26 - Changing the M:N Relationship to Two 1:M Relationships

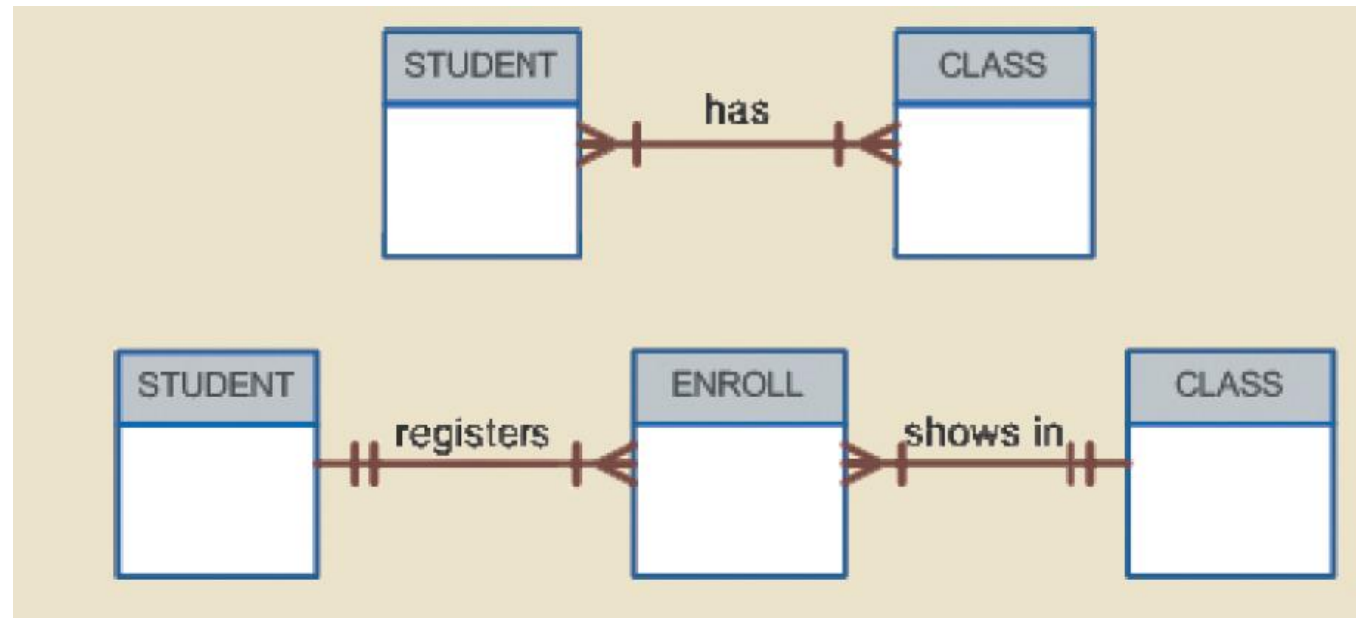
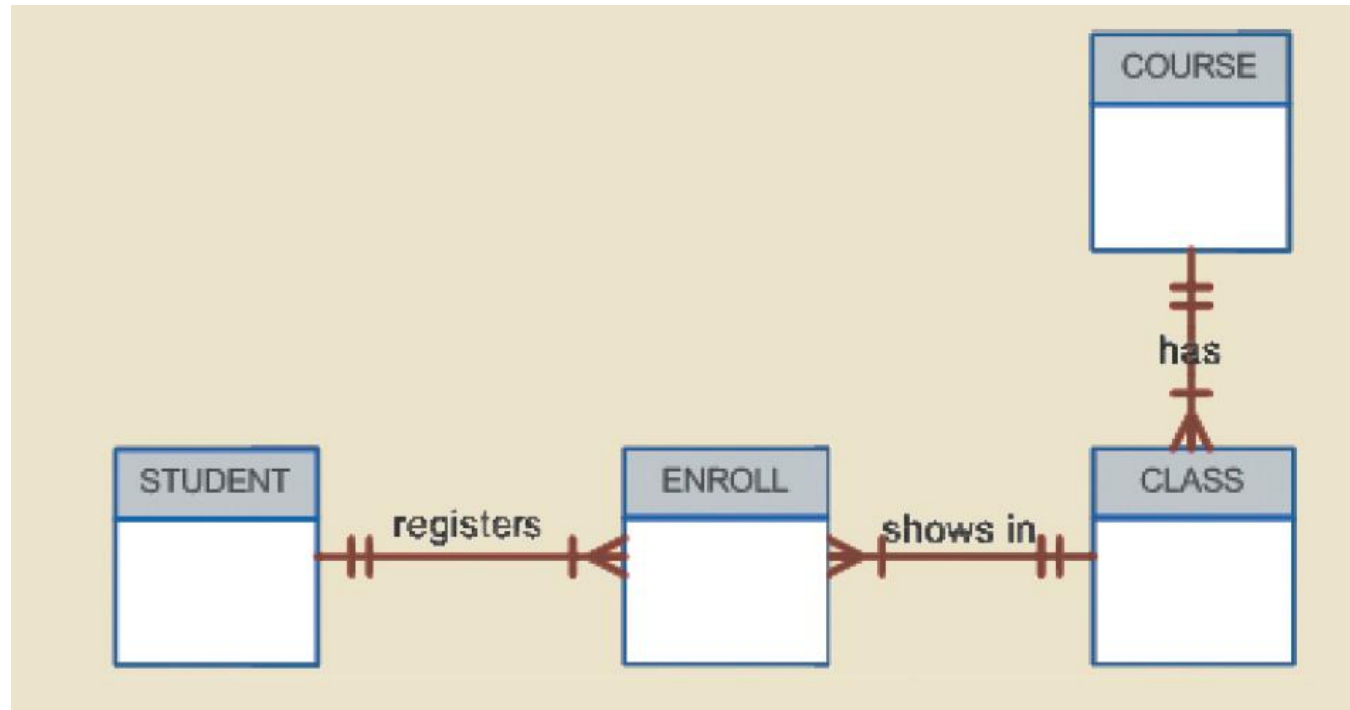


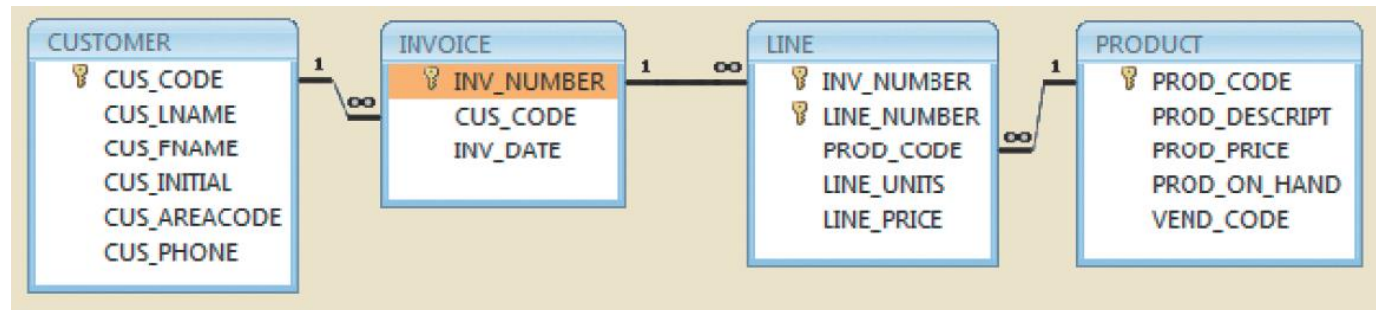
Figure 3.27 - The Expanded ER Model



Data Redundancy Revisited

- 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



Indexes

- 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

Table 3.8 – Dr. Codd’s 12 Relational Database Rules (1 of 2)

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 sub language	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.

Table 3.8 – Dr. Codd's 12 Relational Database Rules (2 of 2)

RULE	RULE NAME	DESCRIPTION
7	High-level insert, update, and delete	The database must support set-level inserts, updates, and deletes.
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.