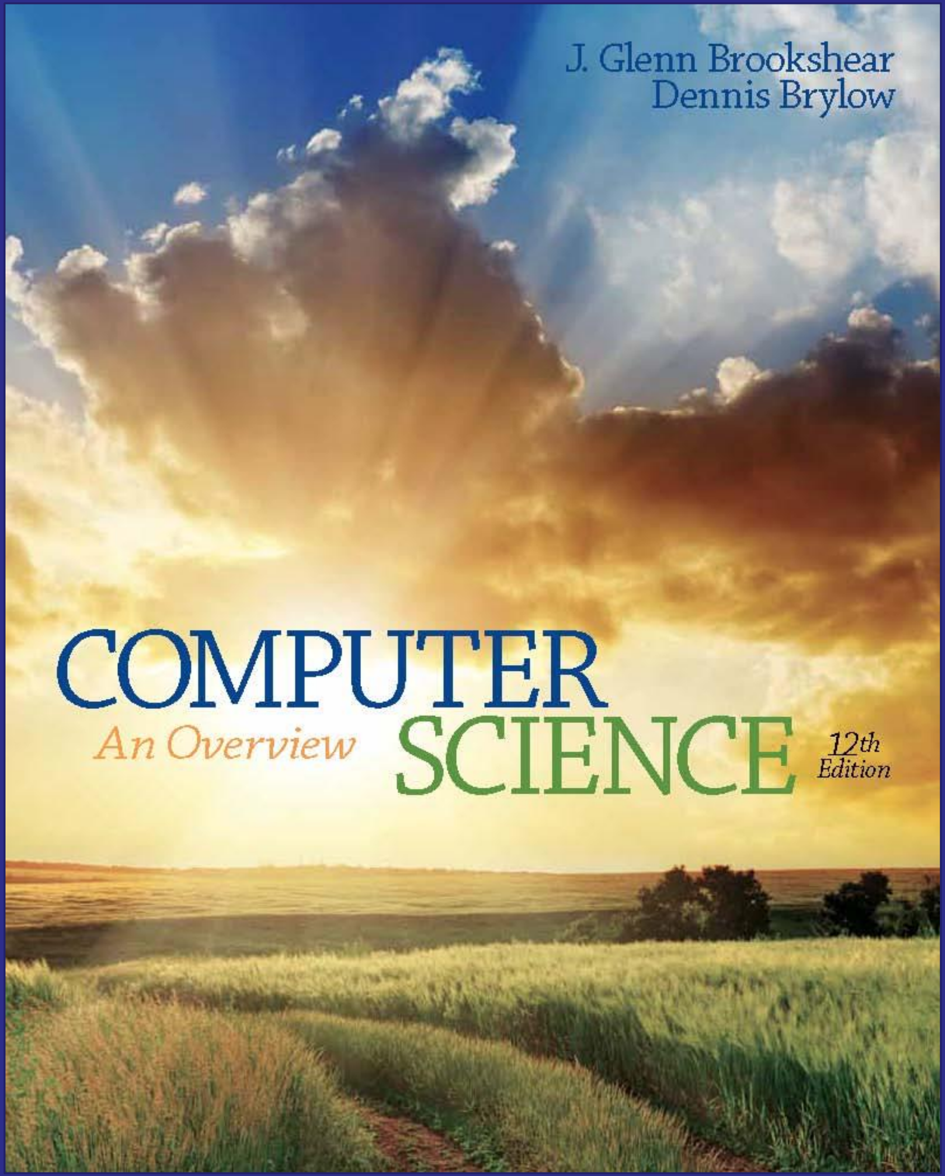
# Chapter 9: Database Systems




PEARSON

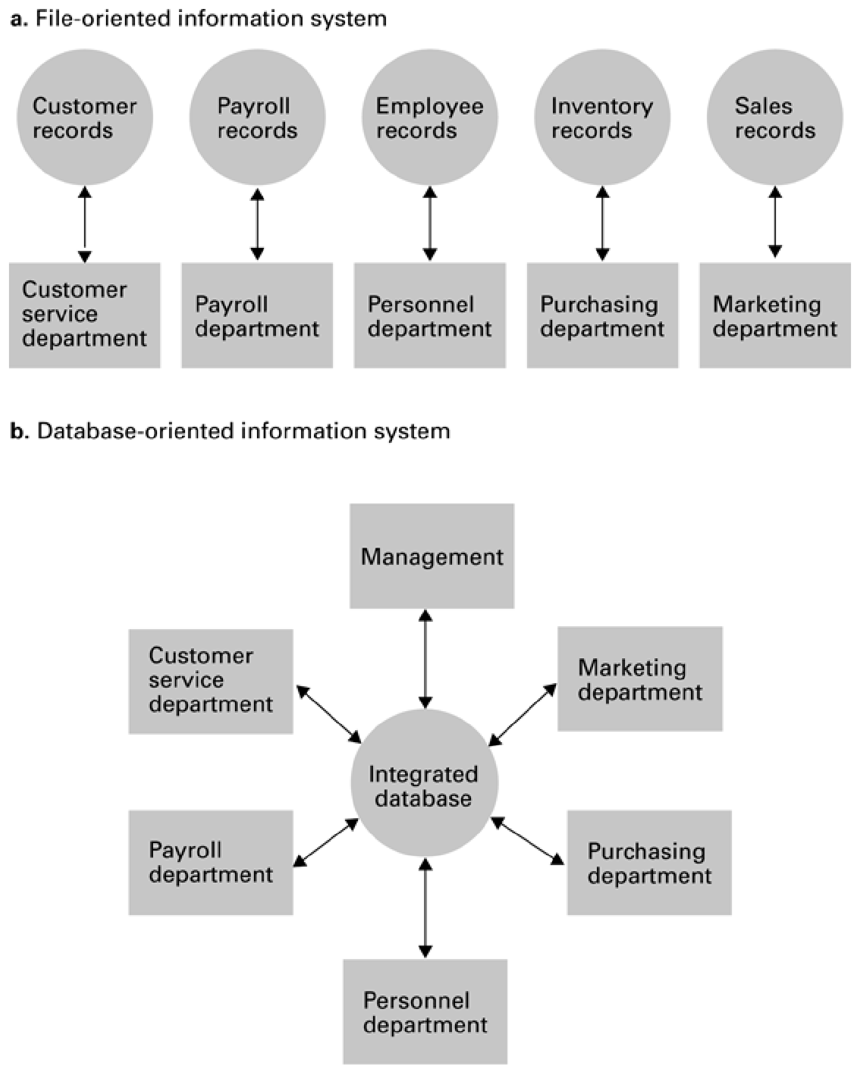
# Chapter 9: Database Systems

- 9.1 Database Fundamentals
- 9.2 The Relational Model
- 9.3 Object-Oriented Databases
- 9.4 Maintaining Database Integrity
- 9.5 Traditional File Structures
- 9.6 Data Mining
- 9.7 Social Impact of Database Technology

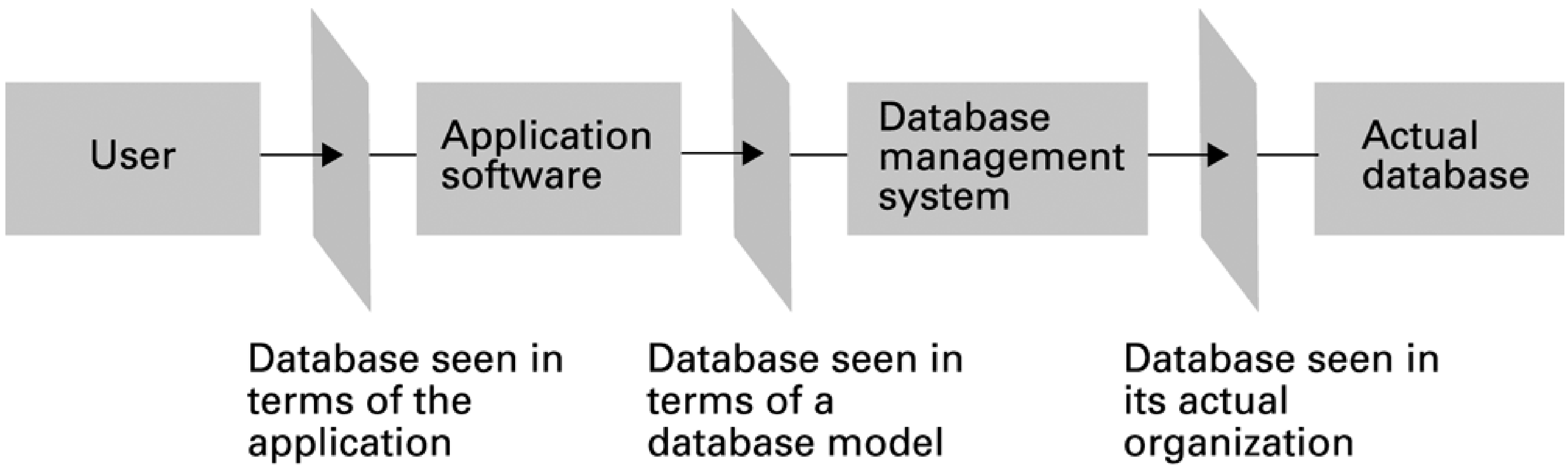
# Database

A collection of data that is multidimensional in the sense that internal links between its entries make the information accessible from a variety of perspectives

# Figure 9.1 **A file versus a database organization**



# Figure 9.2 **The conceptual layers of a database implementation**



# Schemas

- **Schema:** A description of the structure of an entire database, used by database software to maintain the database
- **Subschema:** A description of only that portion of the database pertinent to a particular user's needs, used to prevent sensitive data from being accessed by unauthorized personnel

# Database Management Systems

- **Database Management System** (DBMS): A software layer that manipulates a database in response to requests from applications
- **Distributed Database:** A database stored on multiple machines
  - DBMS will mask this organizational detail from its users
- **Data independence:** The ability to change the organization of a database without changing the application software that uses it

# Database Models

- **Database model:** A conceptual view of a database
  - Relational database model
  - Object-oriented database model

# Relational Database Model

- **Relation:** A rectangular table
  - **Attribute:** A column in the table
  - **Tuple:** A row in the table

# Figure 9.3 **A relation containing employee information**

| Empl Id | Name | Address | SSN |
|---|---|---|---|
| 25X15 | Joe E. Baker | 33 Nowhere St. | 111223333 |
| 34Y70 | Cheryl H. Clark | 563 Downtown Ave. | 999009999 |
| 23Y34 | G. Jerry Smith | 1555 Circle Dr. | 111005555 |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

# Relational Design

- Avoid multiple concepts within one relation
  - Can lead to redundant data
  - Deleting a tuple could also delete necessary but unrelated information

# Improving a Relational Design

- **Decomposition:** Dividing the columns of a relation into two or more relations, duplicating those columns necessary to maintain relationships
  - **Lossless** or **nonloss** decomposition: A "correct" decomposition that does not lose any information

# Figure 9.4 A relation containing redundancy

| Empl Id | Name | Address | SSN | Job Id | Job Title | Skill Code | Dept | Start Date | Term Date |
|---|---|---|---|---|---|---|---|---|---|
| 25X15 | Joe E. Baker | 33 Nowhere St. | 111223333 | F5 | Floor manager | FM3 | Sales | 9-1-2007 | 9-30-2008 |
| 25X15 | Joe E. Baker | 33 Nowhere St. | 111223333 | D7 | Dept. head | K2 | Sales | 10-1-2008 | * |
| 34Y70 | Cheryl H. Clark | 563 Downtown Ave. | 999009999 | F5 | Floor manager | FM3 | Sales | 10-1-2007 | * |
| 23Y34 | G. Jerry Smith | 1555 Circle Dr. | 111005555 | S25X | Secretary | T5 | Personnel | 3-1-1999 | 4-30-2006 |
| 23Y34 | G. Jerry Smith | 1555 Circle Dr. | 111005555 | S26Z | Secretary | T6 | Accounting | 5-1-2006 | * |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

# Figure 9.5 An employee database consisting of three relations

**EMPLOYEE relation**

| Empl Id | Name | Address | SSN |
|---|---|---|---|
| 25X15 | Joe E. Baker | 33 Nowhere St. | 111223333 |
| 34Y70 | Cheryl H. Clark | 563 Downtown Ave. | 999009999 |
| 23Y34 | G. Jerry Smith | 1555 Circle Dr. | 111005555 |

**JOB relation**

| Job Id | Job Title | Skill Code | Dept |
|---|---|---|---|
| S25X | Secretary | T5 | Personnel |
| S26Z | Secretary | T6 | Accounting |
| F5 | Floor manager | FM3 | Sales |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

**ASSIGNMENT relation**

| Empl Id | Job Id | Start Date | Term Date |
|---|---|---|---|
| 23Y34 | S25X | 3-1-1999 | 4-30-2006 |
| 34Y70 | F5 | 10-1-2007 | * |
| 23Y34 | S26Z | 5-1-2006 | * |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

# Figure 9.6 **Finding the departments in which employee 23Y34 has worked**

**EMPLOYEE relation**

| Empl Id | Name | Address | SSN |
|---|---|---|---|
| 25X15 | Joe E. Baker | 33 Nowhere St. | 111223333 |
| 34Y70 | Cheryl H. Clark | 563 Downtown Ave. | 999009999 |
| 23Y34 | G. Jerry Smith | 1555 Circle Dr. | 111005555 |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

**JOB relation**

| Job Id | Job Title | Skill Code | Dept |
|---|---|---|---|
| S25X | Secretary | T5 | Personnel |
| S26Z | Secretary | T6 | Accounting |
| F5 | Floor manager | FM3 | Sales |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

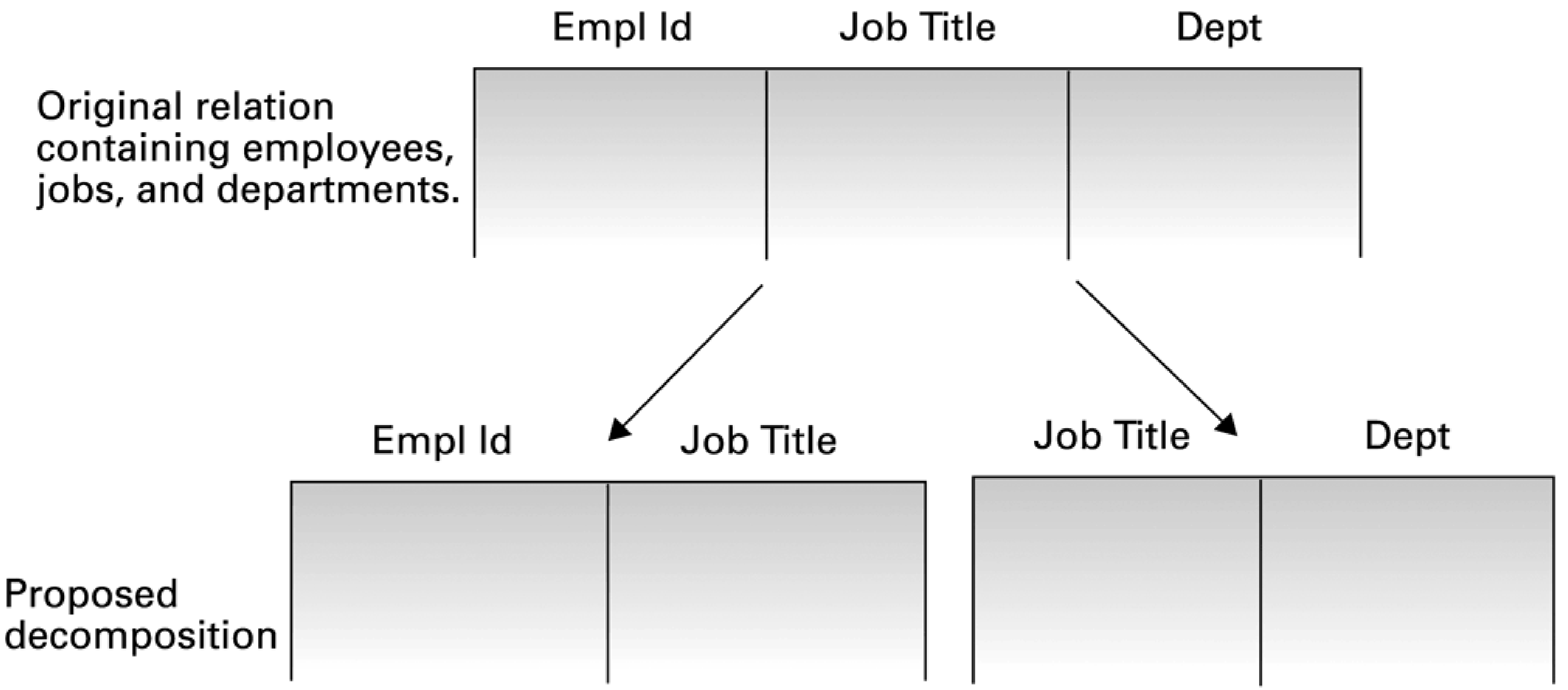
are contained in the personnel and accounting departments.

**ASSIGNMENT relation**

| Empl Id | Job Id | Start Date | Term Date |
|---|---|---|---|
| 23Y34 | S25X | 3-1-1999 | 4-30-2006 |
| 34Y70 | F5 | 10-1-2007 | * |
| 23Y34 | S26Z | 5-1-2006 | * |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

The jobs held by employee 23Y34

# Figure 9.7 **A relation and a proposed decomposition**



# Relational Operations

- **Select:** Choose rows
- **Project:** Choose columns
- **Join:** Assemble information from two or more relations

# Figure 9.8 **The SELECT operation**

**EMPLOYEE relation**

| Empl Id | Name | Address | SSN |
|---|---|---|---|
| 25X15 | Joe E. Baker | 33 Nowhere St. | 111223333 |
| 34Y70 | Cheryl H. Clark | 563 Downtown Ave. | 999009999 |
| 23Y34 | G. Jerry Smith | 1555 Circle Dr. | 111005555 |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

NEW ← SELECT from EMPLOYEE where EmplId = “34Y70”

**NEW relation**

| Empl Id | Name | Address | SSN |
|---|---|---|---|
| 34Y70 | Cheryl H. Clark | 563 Downtown Ave. | 999009999 |

# Figure 9.9 The PROJECT operation

**EMPLOYEE relation**

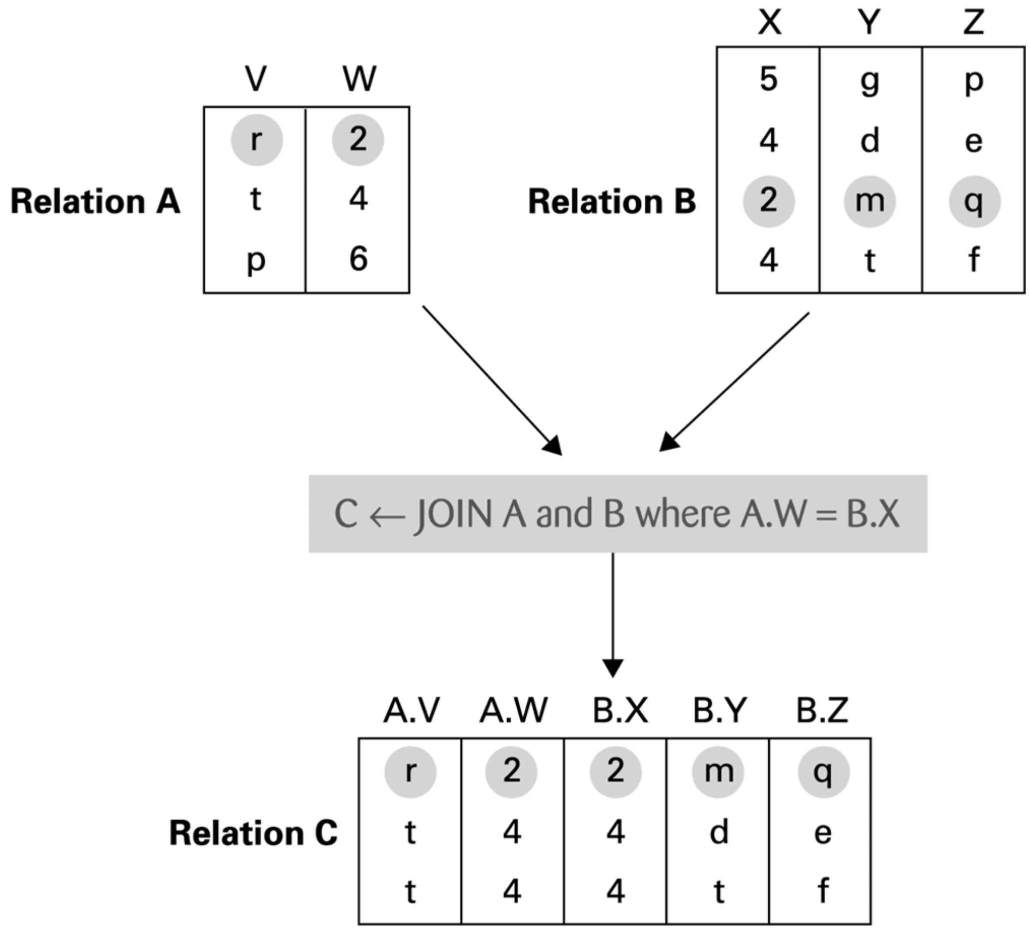
| Empl Id | Name | Address | SSN |
|---|---|---|---|
| 25X15 | Joe E. Baker | 33 Nowhere St. | 111223333 |
| 24Y70 | Cheryl H. Clark | 563 Downtown Ave. | 999009999 |
| 23Y34 | G. Jerry Smith | 1555 Circle Dr. | 111005555 |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

MAIL ← PROJECT Name, Address from EMPLOYEE

**MAIL relation**

| Name | Address |
|---|---|
| Joe E. Baker | 33 Nowhere St. |
| Cheryl H. Clark | 563 Downtown Ave. |
| G. Jerry Smith | 1555 Circle Dr. |
| • | • |
| • | • |
| • | • |

# Figure 9.10 The JOIN operation



# Figure 9.11 **Another example of the JOIN operation**

**Relation A**

| V | W |
|---|---|
| r | 2 |
| t | 4 |
| p | 6 |

**Relation B**

| X | Y | Z |
|---|---|---|
| 1 | g | p |
| 4 | d | e |
| 2 | m | q |
| 5 | t | f |

C ← JOIN A and B where A.W < B.X

**Relation C**

| A.V | A.W | B.X | B.Y | B.Z |
|---|---|---|---|---|
| r | 2 | 4 | d | e |
| r | 2 | 5 | t | f |
| t | 4 | 5 | t | f |

# Figure 9.12 **An application of the JOIN operation**

**ASSIGNMENT relation**

| Empl Id | Job Id | Start Date | Term Date |
|---|---|---|---|
| 23Y34 | S25X | 3-1-1999 | 4-30-2006 |
| 34Y70 | F5 | 10-1-2007 | * |
| 25X15 | S26Z | 5-1-2006 | * |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

**JOB relation**

| Job Id | JobTitle | Skill Code | Dept |
|---|---|---|---|
| S25X | Secretary | T5 | Personnel |
| S26Z | Secretary | T6 | Accounting |
| F5 | Floor manager | FM3 | Sales |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |

NEW1 ← JOIN ASSIGNMENT and JOB where ASSIGNMENT. JobId = JOB.JobId

**NEW1 relation**

| ASSIGNMENT Empl Id | ASSIGNMENT Job Id | ASSIGNMENT StartDate | ASSIGNMENT TermDate | JOB Job Id | JOB JobTitle | JOB SkillCode | JOB Dept |
|---|---|---|---|---|---|---|---|
| 23Y34 | S25X | 3-1-1999 | 4-30-2006 | S25X | Secretary | T5 | Personnel |
| 34Y70 | F5 | 10-1-2007 | * | F5 | Floor manager | FM3 | Sales |
| 25X15 | S26Z | 5-1-2006 | * | S26Z | Secretary | T6 | Accounting |
| • | • | • | • | • | • | • | • |
| • | • | • | • | • | • | • | • |
| • | • | • | • | • | • | • | • |

# Structured Query Language (SQL)

- Operations to manipulate tuples
  - insert
  - update
  - delete
  - select

# SQL Examples

- ```
  SELECT EmplId, Dept
  FROM Assignment, Job
  WHERE Assignment.JobId = Job.JobId
      AND Assignment.TermData = '*';
  ```

- ```
  INSERT INTO Employee
  VALUES ('43212', 'Sue A. Burt',
           '33 Fair St.', '444661111');
  ```

# SQL Examples (continued)

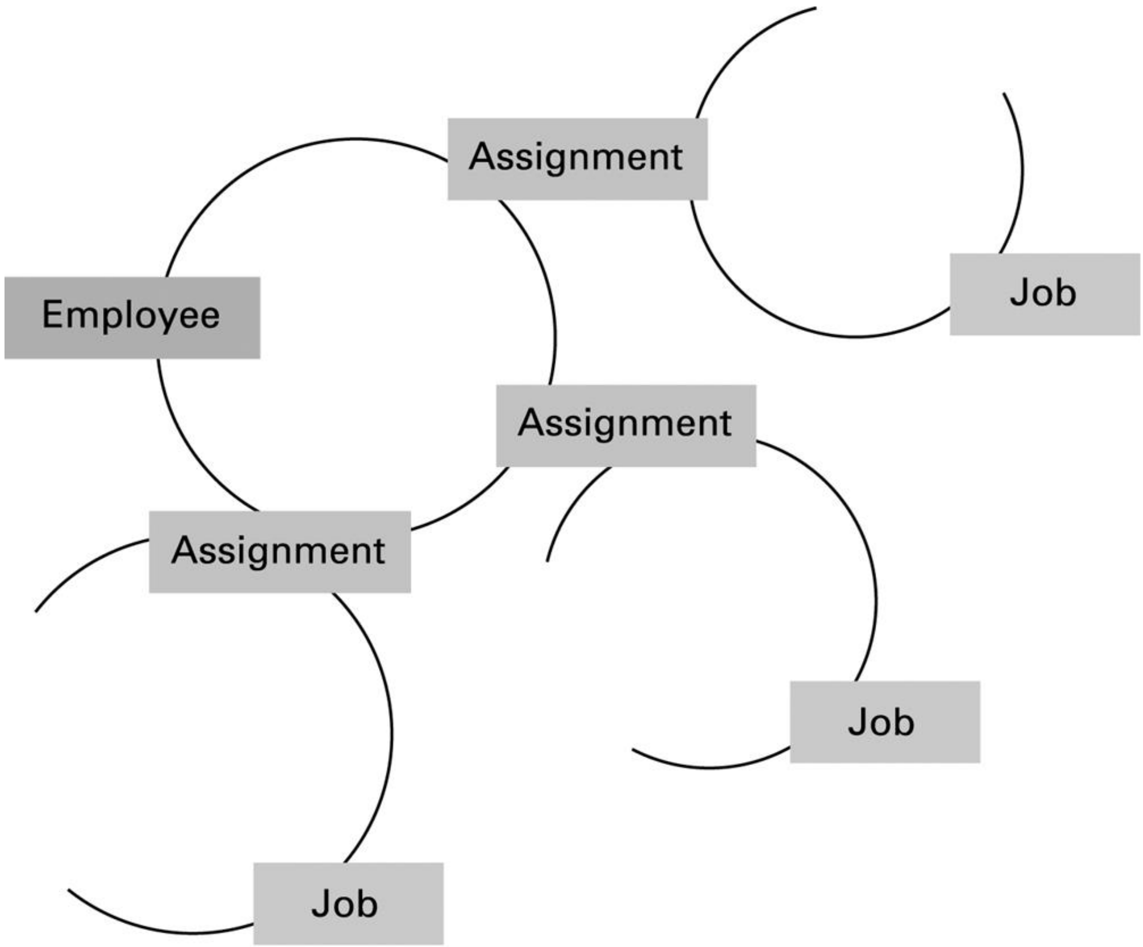
- ```
  DELETE FROM Employee
  WHERE Name = 'G. Jerry Smith';
  ```

- ```
  UPDATE Employee
  SET Address = '1812 Napoleon Ave.'
  WHERE Name = 'Joe E. Baker';
  ```

# Object-oriented Databases

- **Object-oriented Database:** A database constructed by applying the object-oriented paradigm
  - Each entity stored as a persistent object
  - Relationships indicated by links between objects
  - DBMS maintains inter-object links

# Figure 9.13 **The associations between objects in an object-oriented database**



# Advantages of Object-oriented Databases

- Matches design paradigm of object-oriented applications
- Intelligence can be built into attribute handlers
- Can handle exotic data types
  - Example: multimedia

# Maintaining Database Integrity

- **Transaction:** A sequence of operations that must all happen together
  - Example: transferring money between bank accounts
- **Transaction log:** A non-volatile record of each transaction's activities, built before the transaction is allowed to execute
  - **Commit point:** The point at which a transaction has been recorded in the log
  - **Roll-back:** The process of undoing a transaction

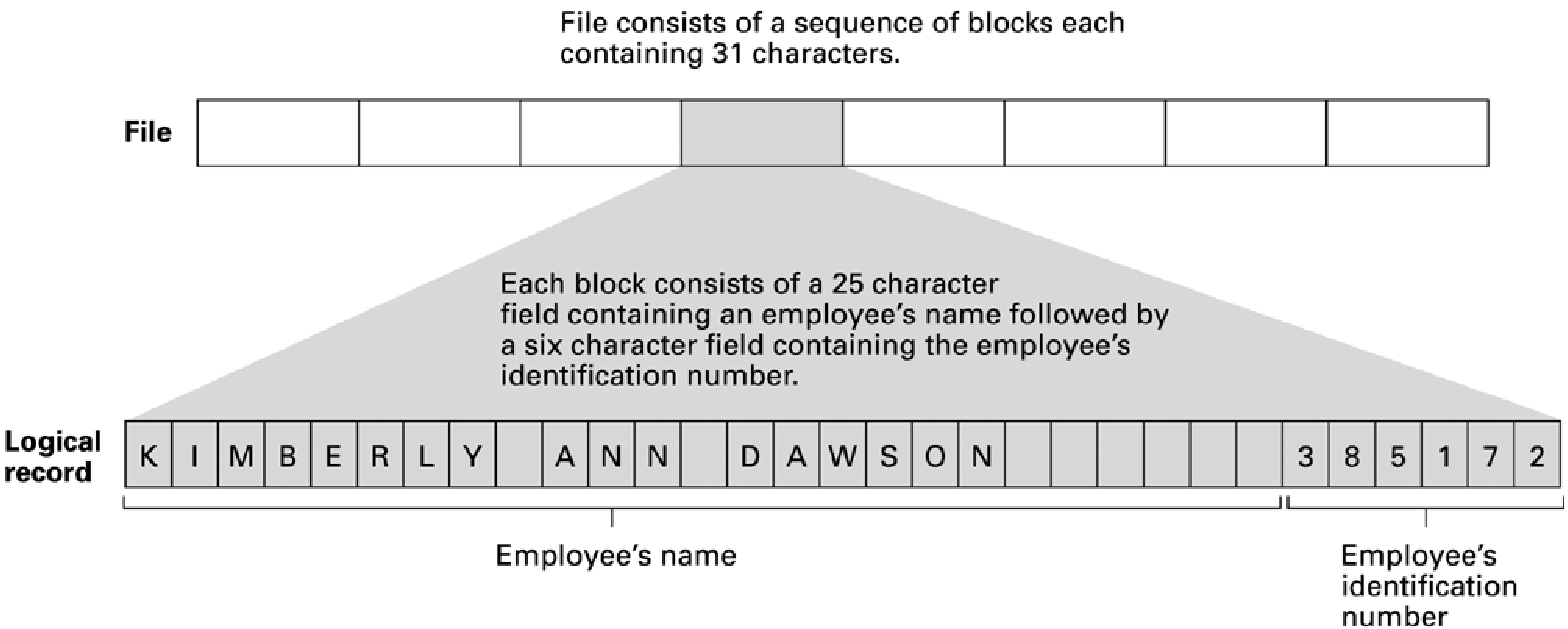
# Maintaining database integrity (continued)

- Simultaneous access problems
  – Incorrect summary problem
  – Lost update problem
- **Locking** = preventing others from accessing data being used by a transaction
  – **Shared** lock: used when reading data
  – **Exclusive** lock: used when altering data

# Sequential Files

- **Sequential file:** A file whose contents can only be read in order
  - Reader must be able to detect end-of-file (EOF)
  - Data can be stored in logical records, sorted by a key field
    - Greatly increases the speed of batch updates

# Figure 9.14 The structure of a simple employee file implemented as a text file

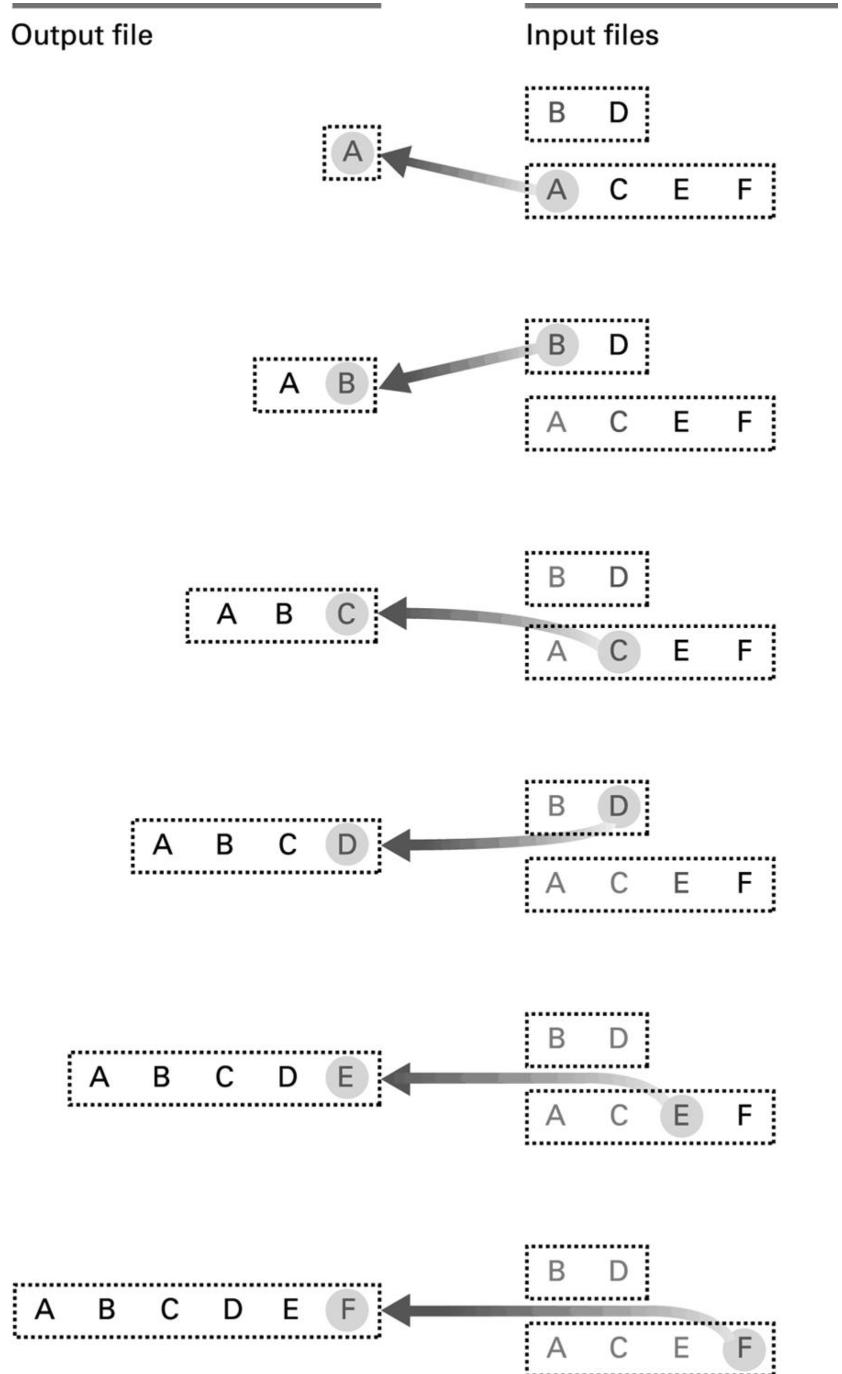


# Figure 9.15 A function for merging two sequential files

```
def MergeFiles (InputFileA, InputFileB, OutputFile):
  if (both input files at EOF):
    Stop, with OutputFile empty

  if (InputFileA not at EOF):
    Declare its first record to be its current record

  if (InputFileB not at EOF):
    Declare its first record to be its current record

  while (neither input file at EOF):
    Put the current record with the "smaller" key field value in OutputFile
    if (that current record is the last record in its corresponding input file):
      Declare that input file to be at EOF
    else:
      Declare the next record in that input file to be the file's current record

  Starting with the current record in the input file that is not at EOF,
  copy the remaining records to OutputFile
```

Figure 9.16

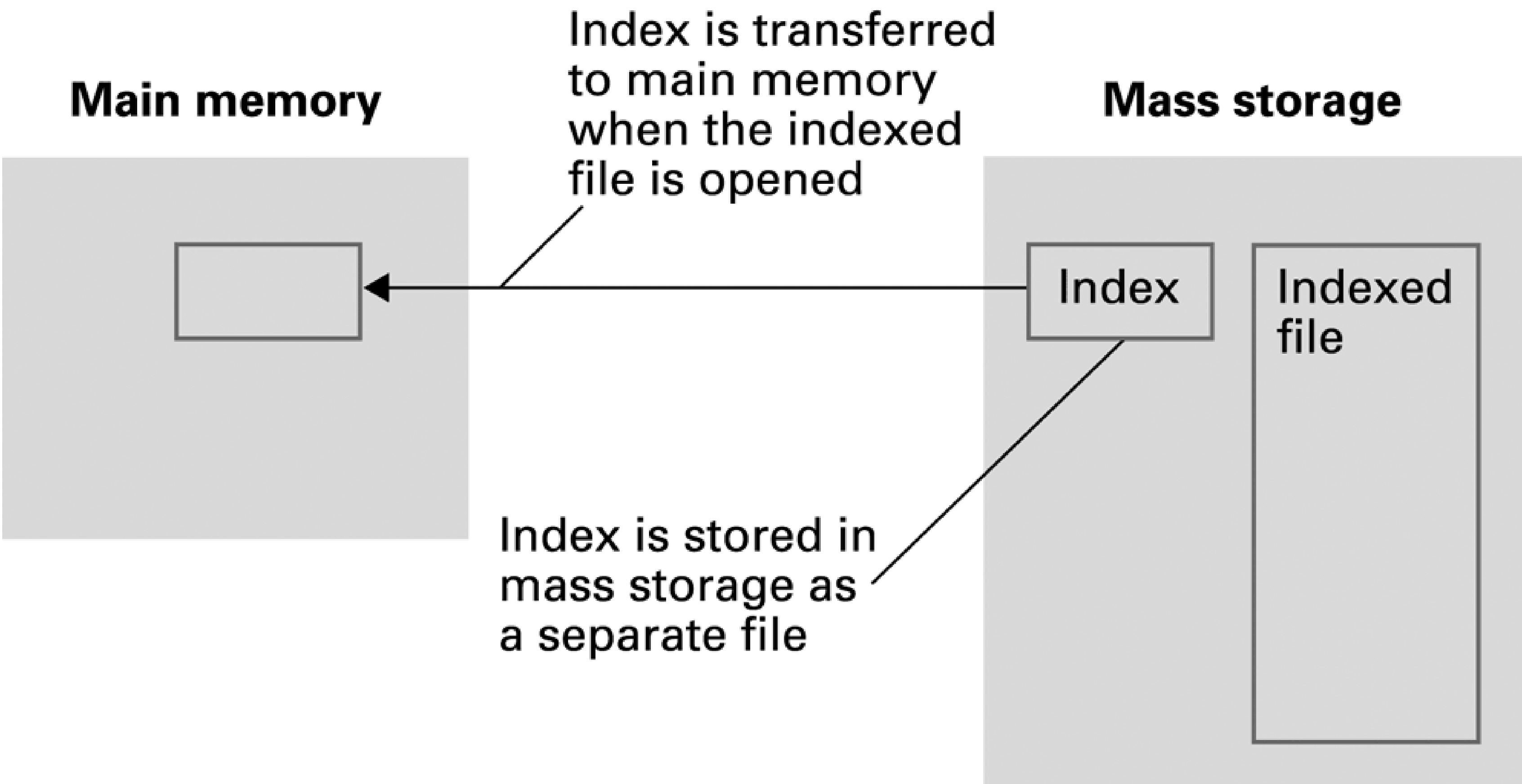
**Applying the merge algorithm (Letters are used to represent entire records. The particular letter indicates the value of the record’s key field.)**



# Indexed Files

- **Index:** A list of key values and the location of their associated records

# Figure 9.17 Opening an indexed file



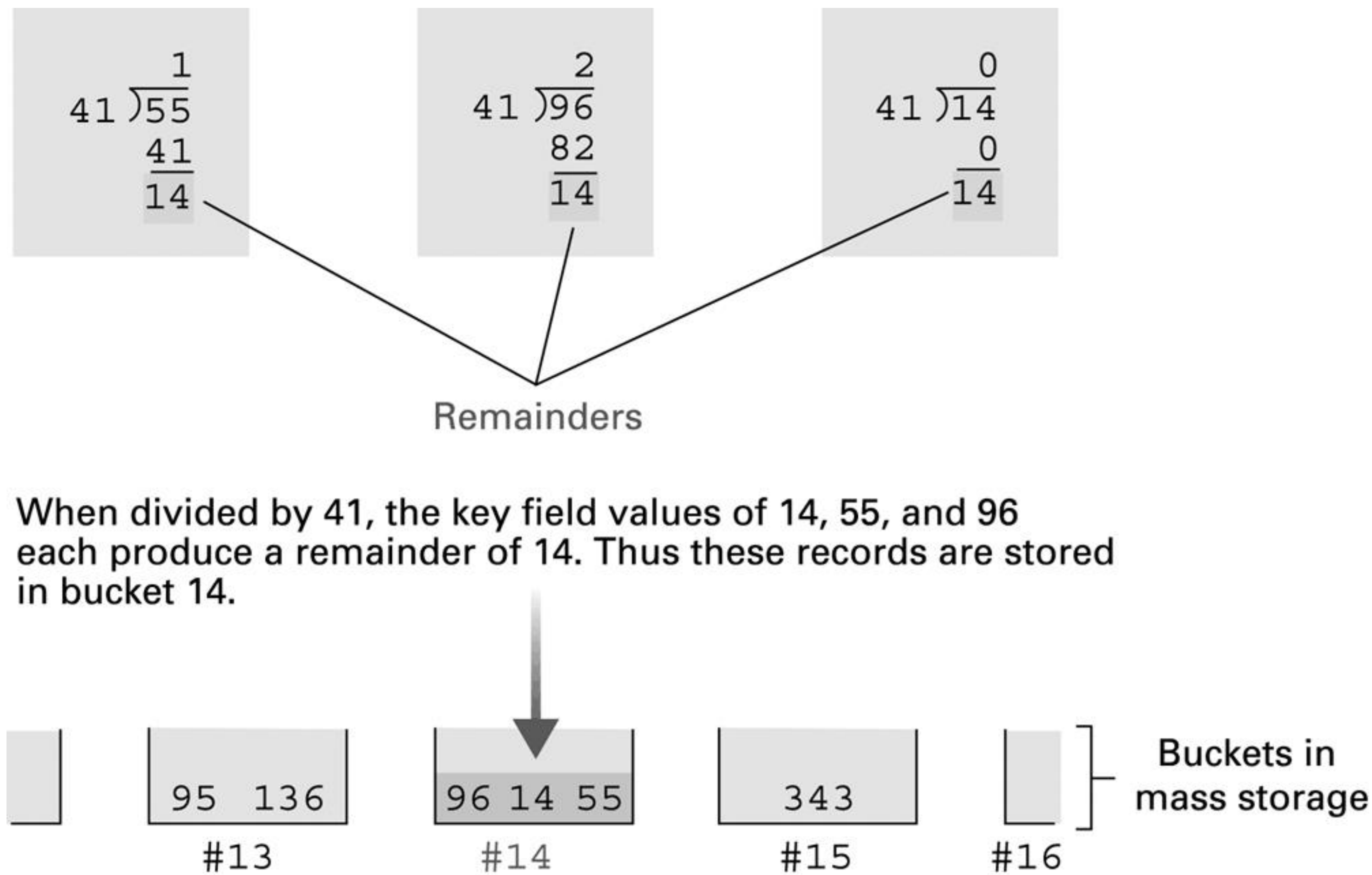
# Hashing

- Each record has a key field
- The storage space is divided into **buckets**
- A **hash function** computes a bucket number for each key value
- Each record is stored in the bucket corresponding to the hash of its key

# Figure 9.18 **Hashing the key field value 25X3Z to one of 41 buckets**

| | |
|---|---|
| Key field value: | 25X3Z |
| ASCII representation: | 0011001000110101010110000011001101011010 |
| Equivalent base ten value: | 215,643,337,562 |
| Remainder after division by 41: | 3 |
| Bucket number: | 3 |

# Figure 9.19 The rudiments of a hashing system



# Collisions in Hashing

- **Collision:** The case of two keys hashing to the same bucket
  - Major problem when table is over 75% full
  - Solution: increase number of buckets and rehash all data

# Data Mining

- **Data Mining:** The area of computer science that deals with discovering patterns in collections of data
- **Data warehouse:** A static data collection to be mined
  - **Data cube:** Data presented from many perspectives to enable mining

# Data Mining Strategies

- Class description
- Class discrimination
- Cluster analysis
- Association analysis
- Outlier analysis
- Sequential pattern analysis

# Social Impact of Database Technology

- Problems
  - Massive amounts of personal data are being collected
    - Often without knowledge or meaningful consent of affected people
  - Data merging produces new, more invasive information
  - Errors are widely disseminated and hard to correct
- Remedies
  - Existing legal remedies often difficult to apply
  - Negative publicity may be more effective

# End
# of
# Chapter