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 9.2** The conceptual layers of a database implementation

- **User**
- **Application software**
- **Database management system**
- **Actual database**

- Database seen in terms of the application
- Database seen in terms of a database model
- Database seen in its actual organization

---

**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

---

**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 Models**

- **Database model**: A conceptual view of a database
  - Relational database model
  - Object-oriented database model
• **Relation:** A rectangular table
  – **Attribute:** A column in the table
  – **Tuple:** A row in the table

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

**Figure 9.3** A relation containing employee information

<table>
<thead>
<tr>
<th>Empl Id</th>
<th>Name</th>
<th>Address</th>
<th>SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>25X15</td>
<td>Joe F. Baker</td>
<td>33 Nowhere St.</td>
<td>111223333</td>
</tr>
<tr>
<td>34Y70</td>
<td>Cheryl H. Clark</td>
<td>563 Downtown Ave.</td>
<td>999009999</td>
</tr>
<tr>
<td>23Y34</td>
<td>G. Jerry Smith</td>
<td>1555 Circle Dr.</td>
<td>11005555</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>EmpId</th>
<th>Name</th>
<th>Address</th>
<th>SSN</th>
<th>JobId</th>
<th>Job Title</th>
<th>Skill Code</th>
<th>Dept</th>
<th>Start Date</th>
<th>Term Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>26X15</td>
<td>Joe E. Baker</td>
<td>33 Nowhere St.</td>
<td>11122333</td>
<td>F5</td>
<td>Floor Manager</td>
<td>FM3</td>
<td>Sales</td>
<td>9-1-2002</td>
<td>9-30-2003</td>
</tr>
<tr>
<td>26X15</td>
<td>Joe E. Baker</td>
<td>33 Nowhere St.</td>
<td>11122333</td>
<td>D7</td>
<td>Dept. Head</td>
<td>K2</td>
<td>Sales</td>
<td>10-1-2003</td>
<td></td>
</tr>
<tr>
<td>34Y70</td>
<td>Cheryl H. Clark</td>
<td>663 Downtown Ave.</td>
<td>11122333</td>
<td>F5</td>
<td>Floor Manager</td>
<td>FM3</td>
<td>Sales</td>
<td>10-1-2002</td>
<td></td>
</tr>
</tbody>
</table>

Figure 9.5 An employee database consisting of three relations

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

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

<table>
<thead>
<tr>
<th>Empl Id</th>
<th>Name</th>
<th>Address</th>
<th>SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>25X15</td>
<td>Joe E. Baker</td>
<td>33 Nowhere St.</td>
<td>111223333</td>
</tr>
<tr>
<td>34Y70</td>
<td>Cheryl H. Clark</td>
<td>563 Downtown Ave.</td>
<td>999009999</td>
</tr>
<tr>
<td>23Y34</td>
<td>G. Jerry Smith</td>
<td>1555 Circle Dr.</td>
<td>111005555</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Empl Id</th>
<th>Name</th>
<th>Address</th>
<th>SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>34Y70</td>
<td>Cheryl H. Clark</td>
<td>563 Downtown Ave.</td>
<td>999009999</td>
</tr>
</tbody>
</table>

---

**Figure 9.9** The PROJECT operation

**Figure 9.10** The JOIN operation

<table>
<thead>
<tr>
<th>Empl Id</th>
<th>Name</th>
<th>Address</th>
<th>SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>25X15</td>
<td>Joe E. Baker</td>
<td>33 Nowhere St.</td>
<td>111223333</td>
</tr>
<tr>
<td></td>
<td>Cheryl H. Clark</td>
<td>563 Downtown Ave.</td>
<td>999009999</td>
</tr>
<tr>
<td></td>
<td>G. Jerry Smith</td>
<td>1555 Circle Dr.</td>
<td>111005555</td>
</tr>
</tbody>
</table>

**MAIL** ← PROJECT Name, Address from EMPLOYEE

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

**Relation A**

<table>
<thead>
<tr>
<th>V</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>2</td>
</tr>
<tr>
<td>t</td>
<td>4</td>
</tr>
<tr>
<td>p</td>
<td>6</td>
</tr>
</tbody>
</table>

**Relation B**

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>g</td>
<td>p</td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>o</td>
</tr>
<tr>
<td>2</td>
<td>m</td>
<td>q</td>
</tr>
<tr>
<td>4</td>
<td>t</td>
<td>f</td>
</tr>
</tbody>
</table>

**C** ← JOIN A and B where A.W = B.X

**Relation C**

<table>
<thead>
<tr>
<th>A.V</th>
<th>A.W</th>
<th>B.X</th>
<th>B.Y</th>
<th>B.Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>4</td>
<td>4</td>
<td>d</td>
<td>4</td>
</tr>
<tr>
<td>t</td>
<td>4</td>
<td>4</td>
<td>e</td>
<td>1</td>
</tr>
</tbody>
</table>
• Operations to manipulate tuples
  – insert
  – update
  – delete
  – select

Structured Query Language (SQL)

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

---

**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

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

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

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

- File consists of a sequence of blocks each containing 31 characters.
- Each block consists of a 25 character field containing an employee’s name followed by a six character field containing the employee’s identification number.
- Employee’s name
- Employee’s identification number
Figure 9.15 A procedure for merging two sequential files

```
procedure MergeFiles (InputFileA, InputFileB, OutputFile)
    if (both input files at EOF) then (Stop, with OutputFile empty)
    if (InputFileA not at EOF) then (Declare its first record to be its current record)
    if (InputFileB not at EOF) then (Declare its first record to be its current record)
    while (neither input file at EOF) do
        if (current record with the "smaller" key field value in OutputFile:
            if (that current record is the last record in its corresponding input file)
                then (Declare that input file to be at EOF)
                else (Declare the next record in that input file to be the file's current record)
            then (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)
                    then (Declare that input file to be at EOF)
                    else (Declare the next record in that input file to be the file's current record)
        else (Copy the remaining records of that input file 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.)

Figure 9.17 Opening an indexed file

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

Indexed Files
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

**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

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

Data Mining Strategies

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