# Chapter 2
# DBMS Concepts and Architecture

- Data Models
- Schemas versus Instances
- Three-Schema Architecture
- Data Independence
- DBMS Languages
- DBMS Interfaces
- DBMS Component Modules
- Database System Utilities
- Classification of DBMSs

## 1. Data Models

- **Data Model**: A set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.
- **Data Model Operations**: Operations for specifying database retrievals and updates by referring to the concepts of the data model
- **Categories of data models:**
  - **Conceptual (high-level, semantic)** data models: Provide concepts that are close to the way many user *perceive* data.(Also called **entity-based** or **object-based** data models.)

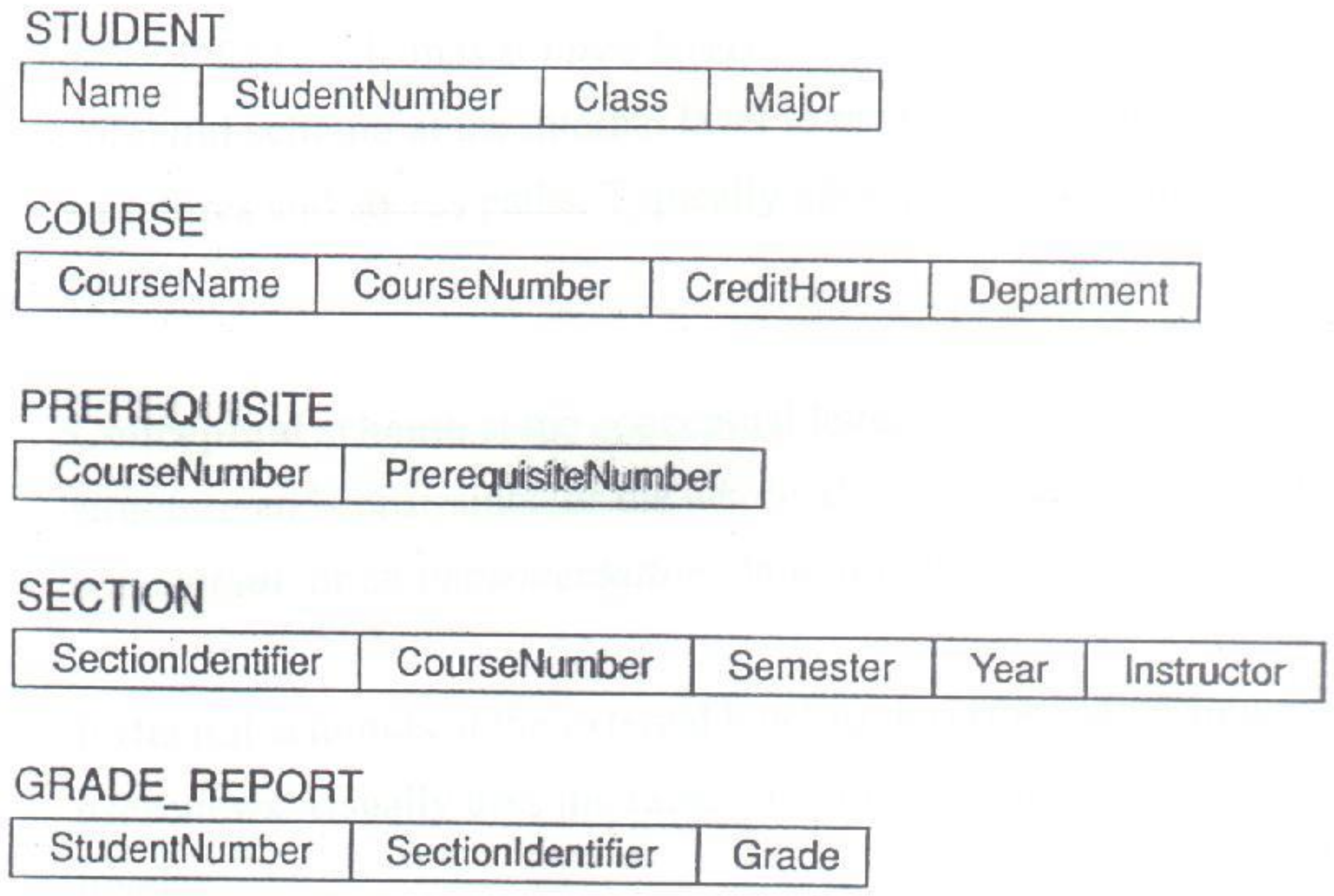
## (cont.)

- **Physical(low-level,internal)** data models: Provide concepts that describe details of how data is stored in the computer.
- **Implementation(record-oriented)** data models: Provide concepts that fall between the above two, balancing user views with some computer storage details.

## 2. Schemas versus Instances

- **Database Schema**: The description of a database. Includes descriptions of the database structure and the constraints that should hold on the database.
- **Schema Diagram**: A diagrammatic display of (some aspects of) a database schema.
- **Database Instance**: The actual data stored in a database at a *particular moment in time*. Also called **database state** (or **occurrence**).
- The **database schema** changes *very infrequently*. The **database state** changes *every time the database is updated.* **Schema** is also called **intension**, whereas **state** is called **extension**.

Figure 2.1



STUDENT

| Name | StudentNumber | Class | Major |
|---|---|---|---|

COURSE

| CourseName | CourseNumber | CreditHours | Department |
|---|---|---|---|

PREREQUISITE

| CourseNumber | PrerequisiteNumber |
|---|---|

SECTION

| SectionIdentifier | CourseNumber | Semester | Year | Instructor |
|---|---|---|---|---|

GRADE_REPORT

| StudentNumber | SectionIdentifier | Grade |
|---|---|---|

# 3. Three-Schema Architecture

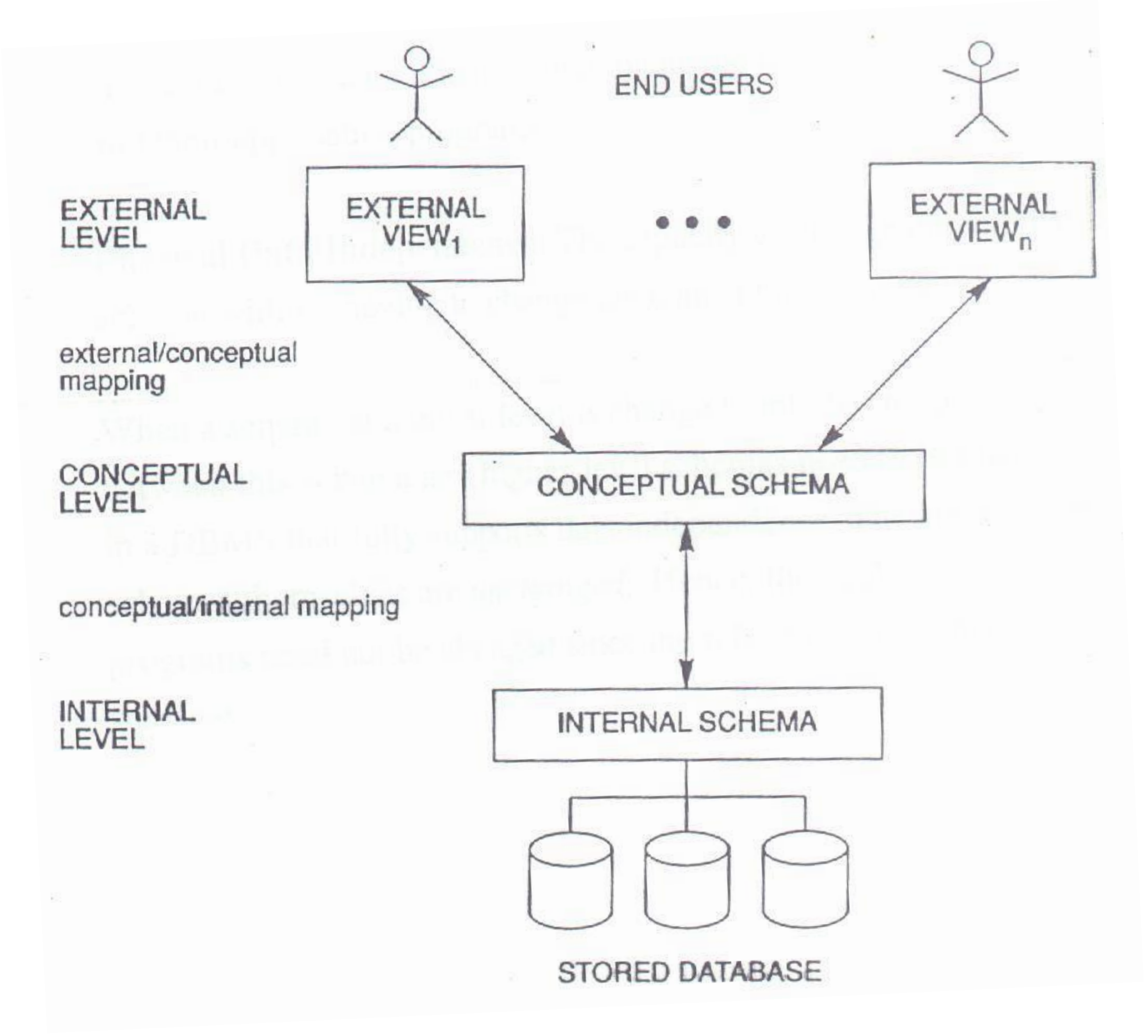
- Proposed to support DBMS characteristics of:
  - **Program-data independence**
  - Support of **multiple views** of the data.
- Defines DBMS schemas at *three levels*:
  - **Internal schema** at the internal level to describe data storage structures and access paths. Typically uses a *physical* data model.
  - **Conceptual schema** at the conceptual level to describe the structure and constraints for the *whole* database. Uses a *conceptual* or an *implementation* data model.

# (cont.)

- **External schemas** at the external level to describe the various user views. Usually uses the same data model as the conceptual level.

- **Mappings** among schema levels are also needed. Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.

Figure 2.2


END USERS
EXTERNAL LEVEL
EXTERNAL VIEW$_1$
• • •
EXTERNAL VIEW$_n$
external/conceptual mapping
CONCEPTUAL LEVEL
CONCEPTUAL SCHEMA
conceptual/internal mapping
INTERNAL LEVEL
INTERNAL SCHEMA
STORED DATABASE

## 4. Data Independence

- **Logical Data Independence**: The capacity to change the conceptual schema without having to change the external schemas and their application programs.
- **Physical Data Independence**: The capacity to change the internal schema without having to change the conceptual schema.
- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.The higher-level schemas themselves are *unchanged*. Hence, the application programs need not be changed since the refer to the external schemas.

## 5. DBMS Languages

- **Data Definition Language(DDL)**: Used by the DBA and database designers to specify the *conceptual schema* of a database.In many DBMSs, the DDL is also used to define internal and external schemas (views). In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.
- **Data Manipulation Language(DML)**: used to specify database retrievals and updates

## (cont.)

- DML commands **(data sublanguage**) can be *embedded* in a general-purpose programming language**(host language)**, such as COBOL, PL/1 or PASCAL.
- Alternatively, *stand-alone* DML commands can be applied directly **(query language**)

## Types of DML

- **Procedural DML**: Also called *record-at-a-time* or low-level DML. Must be embedded in a programming language. Searches for and retrieves individual database records, and uses looping and other constructs of the host programming language to retrieve multiple records.
- **Declarative or non-procedural DML**: Also called *set-at-a-time* or *high-level* DML. Can be used as a *stand-alone* query language or can be *embedded* in a programming language. Searches for and retrieves information from *multiple* related database records in a single command.

## 6. DBMS Interfaces

- Stand-alone query language interfaces
- Programmer interfaces for embedding DML in programming languages:
  - Pre-compiler Approach
  - Procedure(Subroutine) Call Approach
- User-friendly interfaces:
  - Menu-based
  - Graphics-based
  - Forms-based
  - Natural language
  - Combinations of the above

## (cont.)

- Parametric interfaces using function keys
- Report generation languages
- Interfaces for the DBA:
  - Creating accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access path

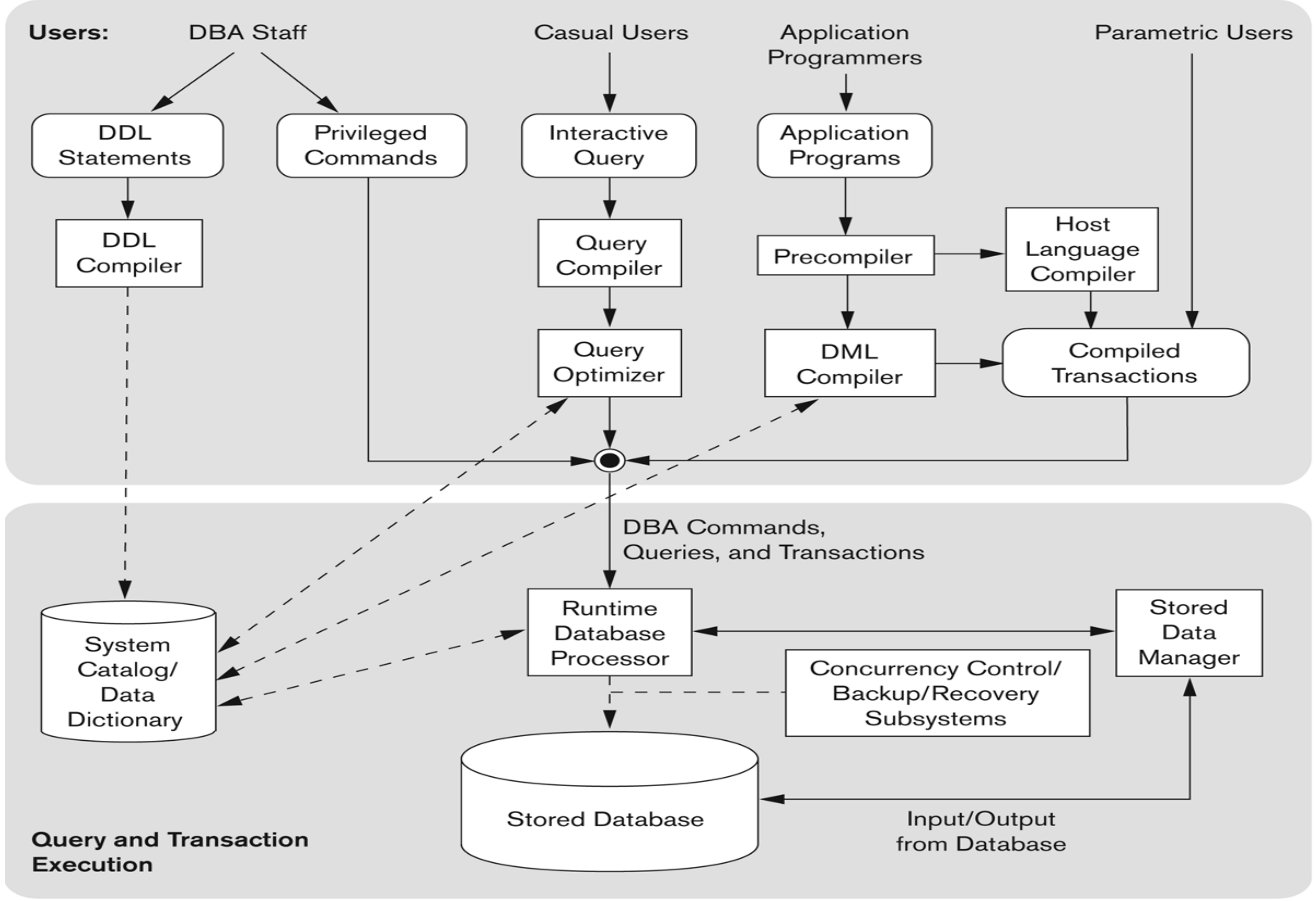


**Figure 2.3**
Component modules of a DBMS and their interactions.

## 8. Database System Utilities

- To perform certain functions such as:
  - *Loading* data stored in files into a database
  - *Backing up* the database periodically on tape
  - *Reorganizing* database file structures
  - *Report generation* utilities
  - *Performance monitoring* utilities
  - Other functions, such as *sorting,user monitoring*, *data compression*,etc.
- Data dictionary utility:
  - Used to store schema descriptions and other information such as design decisions, application program descriptions,user information,usage standards,etc.

## (cont.)

- *Active* data dictionary is accessed by DBMS software and users/DBA
- *Passive* data dictionary is accessed by users/DBA only.

## 9. Classification of DBMSs

- Based on the data model used:
  - Traditional: Relational, Network, Hierarchical
  - Emerging: Object-oriented, Semantic, Entity-Relationship, other
- Other classifications:
  - Single-user (typically used with micro-computers) vs. multi-user (most DBMSs)
  - Centralized (uses a single computer) vs. distributed (uses multiple computers)
  - Cost of DBMS software
  - Types of access paths used.