Database System Concepts and Architecture

- **Data Models:** A data model a collection of concepts that can be used to describe the structure of a database provides the necessary means to achieve this abstraction, By structure of a database we mean the data types, relationships, and constraints that apply to the data.

Categories of Data Models:

1) High-level or conceptual data models:

- Provide concepts that are close to the way many users perceive data**.**
- Conceptual data models use concepts such as entities, attributes, and relationships.
- An entity represents a real-world object or concept, such as an employee or a project from the miniworld that is described in the database.
- An attribute represents some property of interest that further describes an entity, such as the employee's name or salary.
- A relationship among two or more entities represents an association among the entities**.**

2) Low-level or physical data models:

- Provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks.
- Concepts provided by physical data models are generally meant for computer specialists, not for end users.

3) Representational (or implementation) data models:

- Which provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage.
- Representational data models hide many details of data storage on disk but can be implemented on a computer system directly.
- Representational or implementation data models are the models used most frequently in traditional commercial DBMSs.

- **Schema:** The description of a database is called the Database schema.
  - This is specified during database design and is not expected to change frequently. Most data models have certain conventions for displaying schemas as diagrams. A displayed schema is called a schema diagram.

Figure 2.1
Schema diagram for the database in Figure 1.2.

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

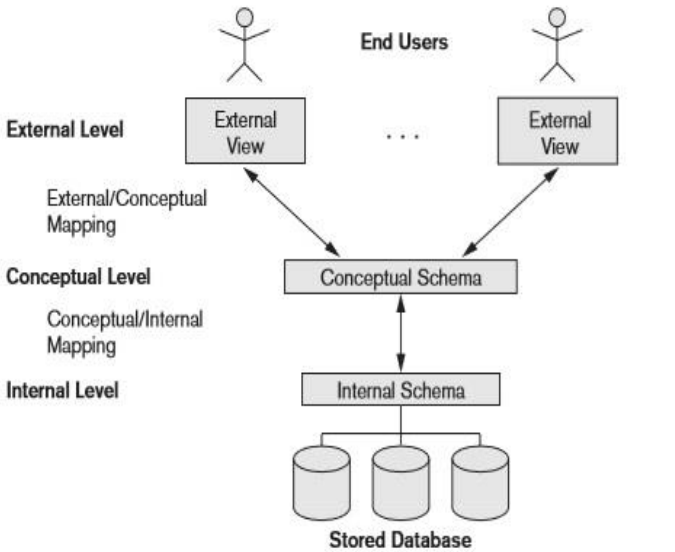
GRADE_REPORT

Student_number	Section_identifier	Grade

- **Instances:** The data in the database at a particular moment in time is called a database state or snapshot. It is also called the current set of occurrences or instances in the database.
  - **Example:** customer name is schema and George Grant is an instance of customer name.

- **Three-Schema Architecture:**
  - The goal of the three-schema architecture, is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

Figure 2.2
The three-schema architecture.



1. **The internal level has an internal schema**:
   - This describes the physical storage structure of the database.
   - The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
2. **The conceptual level has a conceptual schema**:
   - This describes the structure of the whole database for a community of users.
   - The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
   - Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.
3. **The external or view level:**
   - Includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.
   - As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level conceptual data model.

❖ **Data Independence:**

- Data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. We can define two types of data independence:

1. **Logical data independence:**

- Is the capacity to change the conceptual schema without having to change external schemas or application programs.
- We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).

2. **Physical data independence:**

- is the capacity to change the internal schema without having to change the conceptual schema.
- Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized.

❖ Database Languages:

1. Data Definition Language(DDL):

- Specify conceptual and internal schemas for the database and any mappings between the two.
- In many DBMSs where no strict separation of levels is maintained, one language, called the data definition language (DDL), is used by the DBA and by database designers to define both schemas.
- The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.
- In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only.

2. Storage definition language (SDL):

- Is used to specify the internal schema. The mappings between the two schemas may be specified in either one of these languages.
- In most relational DBMSs today, there is no specific language that performs the role of SDL. Instead, the internal schema is specified by a combination of functions, parameters, and specifications related to storage of files.
- These permit the DBA staff to control indexing choices and mapping of data to storage.

3. View definition language (VDL):

- To specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.
- In relational DBMSs, SQL is used in the role of VDL to define user or application views as results of predefined queries.

4. Data manipulation language (DML):

- The database schemas are compiled and the database is populated with data, users must have some means to manipulate the database.
- Typical manipulations include retrieval, insertion, deletion, and modification of the data. The DBMS provides a set of operations or a language called the data manipulation language (DML) for these purposes.
- There are **two main types of DMLs**.
- **A high-level or nonprocedural DML** can be used on its own to specify complex database operations concisely. Many DBMSs allow high-level DML statements either to be entered interactively from a display monitor or terminal or to be embedded in a general-purpose programming language.

- **A low-level or procedural** DML must be embedded in a general-purpose programming language. This type of DML typically retrieves individual records or objects from the database and processes each separately.

❖ DBMS Interfaces:

1) **Menu-based Interfaces for Web Clients or Browsing**:
   - These interfaces present the user with lists of options (called menus) that lead the user through the formulation of a request. Menus do away with the need to memorize the specific commands and syntax of a query language
2) **Apps for Mobile Devices**:
   - These interfaces present mobile users with access to their data. For example, banking, reservations, and insurance companies, among many others, provide apps that allow users to access their data through a mobile phone or mobile device.
3) **Forms-based Interfaces**:
   - A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries. Forms are usually designed and programmed for naive users as interfaces to canned transactions. Many DBMSs have forms specification languages, which are special languages that help programmers specify such forms. SQL*Forms is a form-based language that specifies queries using a form designed in conjunction with the relational database schema.
4) **Graphical User Interfaces**:
   - A GUI typically displays a schema to the user in diagrammatic form. The user then can specify a query by manipulating the diagram. In many cases, GUIs utilize both menus and forms.
5) **Natural Language Interfaces**:
   - These interfaces accept requests written in English or some other language and attempt to understand them. A natural language interface usually has its own schema, which is similar to the database conceptual schema, as well as a dictionary of important words.
6) **Keyword-based Database Search**:
   - These are somewhat similar to Web search engines, which accept strings of natural language (like English or Spanish) words and match them with documents at specific sites (for local search engines) or Web pages on the Web at large (for engines like Google or Ask).
7) **Speech Input and Output**:
   - Limited use of speech as an input query and speech as an answer to a question or result of a request is becoming commonplace. Applications with limited vocabularies, such as inquiries for telephone directory, flight arrival/departure, and

credit card account information, are allowing speech for input and output to enable customers to access this information.

8) **Interfaces for the DBA**:

- Most database systems contain privileged commands that can be used only by the DBA staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

The Database System Environment

❖ DBMS Component Modules:

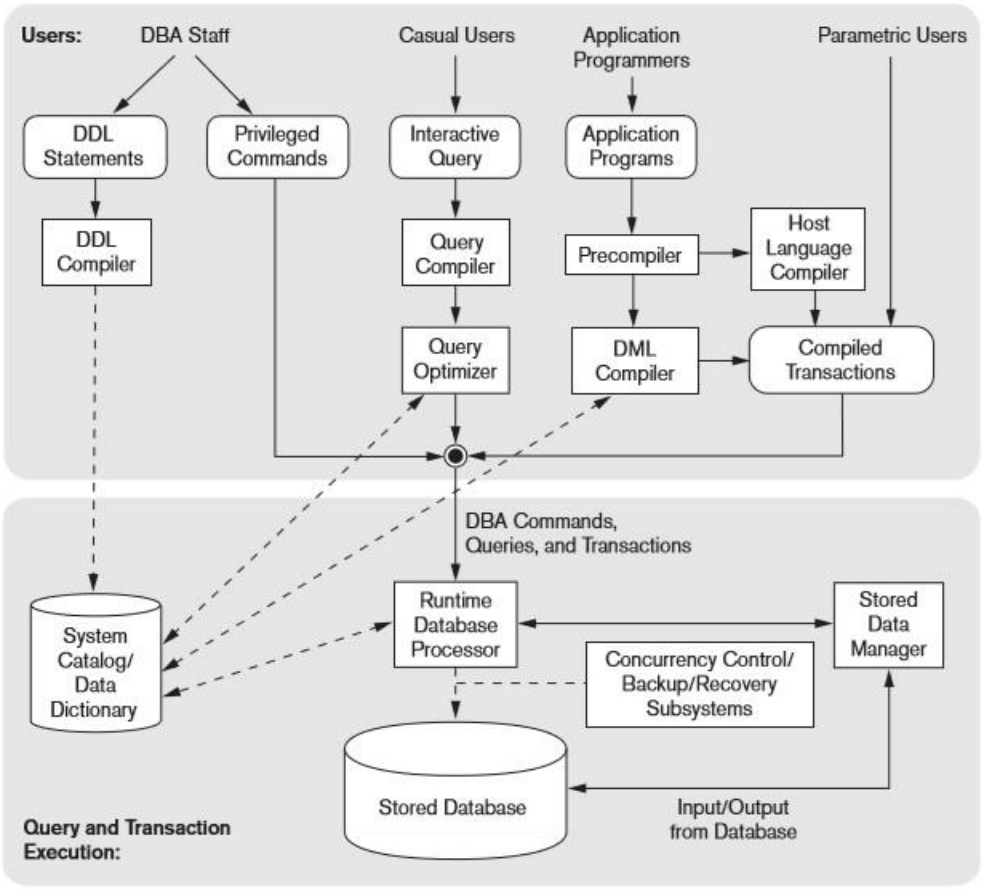


Figure 2.3
Component modules of a DBMS and their interactions.

- The figure is divided into two halves. The top half of the figure refers to the various users of the database environment and their interfaces. The lower half shows the internals of the DBMS responsible for storage of data and processing of transaction.

- The database and the DBMS catalog are usually stored on disk. Access to the disk is primarily controlled by operating system(OS).which includes disk input/output. A higher level stored data manager module of DBMS controls access to DBMS information that is stored on the disk.
- If we consider the top half of the figure, It shows interfaces to DBA staff, casual users, application programmers and parametric users.
- The DDL compiler processes schema definitions, specified in the DDL, and stores the description of the schema in the DBMS Catalog..The catalog includes information such as names and sizes of the sizes of the files, data types of data of data items. Storage details of each file, mapping information among schemas and constraints.
- Casual users and persons with occasional need of information from database interact using some for of interface which is interactive query interface. The queries are parsed, analysed for correctness of the operations for the model.
- the names of the data elements and so on by a query compiler that compiles them into internal form. The internal query is subjected to query optimization..The query optimizer is concerned with rearrangement and possible recording of operations, eliminations of redundancies.
- Application programmer writes programs in host languages. The precompiled extracts DML commands from an application program.

❖ Database System Utilities:

■ Loading:

- A loading utility is used to load existing data files—such as text files or sequential files—into the database.
- Usually, the current (source) format of the data file and the desired (target) database file structure are specified to the utility, which then automatically reformats the data and stores it in the database.
- With the proliferation of DBMSs, transferring data from one DBMS to another is becoming common in many organizations.
- Some vendors offer conversion tools that generate the appropriate loading programs, given the existing source and target database storage descriptions (internal schemas).

■ Backup:

- A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium.
- The backup copy can be used to restore the database in case of catastrophic disk failure.

- Incremental backups are also often used, where only changes since the previous backup are recorded.
- Incremental backup is more complex, but saves storage space.

■ Database storage reorganization:

- This utility can be used to reorganize a set of database files into different file organizations and create new access paths to improve performance.

■ Performance monitoring:

- Such a utility monitors database usage and provides statistics to the DBA.
- The DBA uses the statistics in making decisions such as whether or not to reorganize files or whether to add or drop indexes to improve performance.

❖ Centralized and Client/Server Architectures for DBMSs:

Centralized DBMSs Architecture:

- Architectures for DBMSs have followed trends similar to those for general computer system architectures.
- Older architectures used mainframe computers to provide the main processing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality.
- The reason was that in older systems, most users accessed the DBMS via computer terminals that did not have processing power and only provided display capabilities.
- Therefore, all processing was performed remotely on the computer system housing the DBMS, and only display information and controls were sent from the computer to the display terminals, which were connected to the central computer via various types of communications networks.
- As prices of hardware declined, most users replaced their terminals with PCs and workstations, and more recently with mobile devices.
- At first, database systems used these computers similarly to how they had used display terminals, so that the DBMS itself was still a centralized DBMS in which all the DBMS functionality application program execution

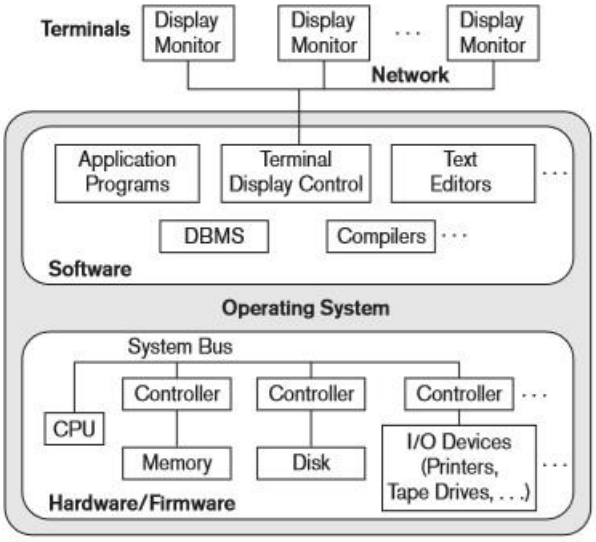


Figure 2.4
A physical centralized architecture.

Basic Client/Server Architectures:

- The client/server architecture was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, database servers, Web servers, e-mail servers, and other software and equipment are connected via a network.
- The idea is to define specialized servers with specific functionalities. For example, it is possible to connect a number of PCs or small workstations as clients to a file server that maintains the files of the client machines.
- Another machine can be designated as a printer server by being connected to various printers; all print requests by the clients are forwarded to this machine.
- Web servers or e-mail servers also fall into the specialized server category. The resources provided by specialized servers can be accessed by many client machines.
- The client machines provide the user with the appropriate interfaces to utilize these servers, as well as with local processing power to run local applications.

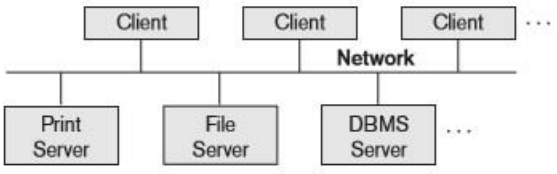


Figure 2.5
Logical two-tier client/server architecture.

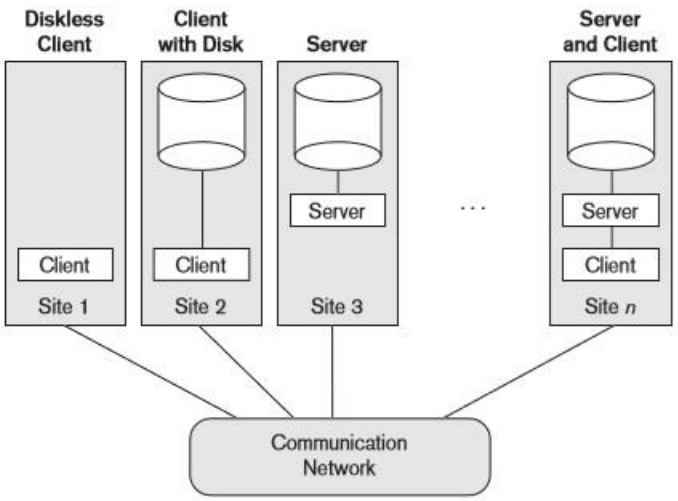


Figure 2.6
Physical two-tier client/server architecture.

- The concept of client/server architecture assumes an underlying framework that consists of many PCs/workstations and mobile devices as well as a smaller number of server machines, connected via wireless networks or LANs and other types of computer networks.
- A client in this framework is typically a user machine that provides user interface capabilities and local processing.
- A server is a system containing both hardware and software that can provide services to the client machines, such as file access, printing, archiving, or database access.

❖ Classification of Database Management Systems

- Homogeneous
- Heterogeneous

Homogeneous Distributed Database:

- In a homogeneous distributed database, all sites have identical management system software that agrees to cooperate in processing users requests.
- In such a system, local sites surrender a portion of their autonomy in terms of their right to change schemes or DBMS software. .
- Homogeneous DDBS is the simplest form of a distributed database where there are several sites, each running their own applications on the same DBMS software.
- All sites have identical DBMS software, all users use identical software, are aware of one another end agree to co-operate in processing user's request.
- The application can all see the same schema and run the same transactions. That is, there is location transparency in homogeneous DDBS.

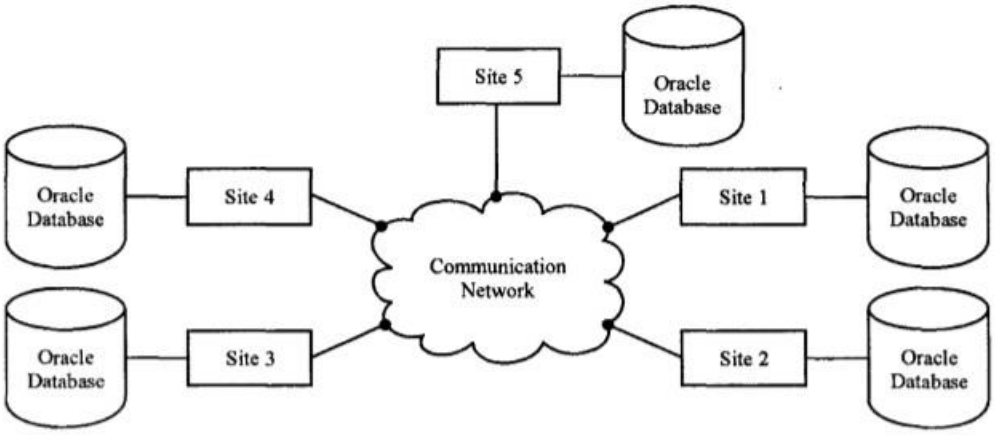


Fig. Homogeneous distributed database architecture.

Heterogeneous Distributed Database:

- In a heterogeneous distributed database, different sites may use different schemes and different database management system software.
- This site may not be aware of one another and they may provide only limited facilities for cooperation in transaction processing.

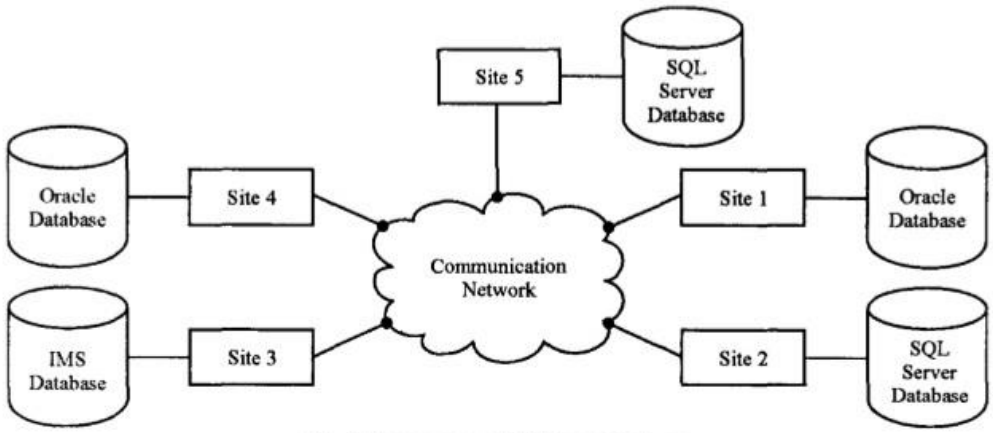


Fig. Heterogeneous DDBS acrhitecture.

- Heterogeneous distributed database system is also referred to as a multi-database system or a federated database system.
- Heterogeneous database systems have well-accepted standards for gateway protocols to expose DBMS functionality to external applications.
- The gateway protocols help in masking the differences of accessing database servers, and bridge the differences between the different servers in a distributed system.