CS3CRT06 : Database Management Systems (Core)

Syllabus: Unit I: Introduction
Characteristics of the Database Approach - Database users :DBA , Database Designers End users - Advantages of using the DBMS Approach - Data models. Schemas, and Instances - Three-Schema Architecture and Data Independence.
DBMS Languages: DDL, DML - The Database System Environment: DBMS Component Modules.

Database Management Systems

Data: Data means facts that can be recorded.
Examples: names, telephone numbers, addresses, etc.
(names, numbers, images, audios, videos are all data)

Information : Processed data is called information.
Examples: Telephone Directory, Mark List, Electricity Bill, etc.

Database: A database is a collection of related data.
For example, when the names, telephone numbers, and addresses of the people you know are stored in a computer and that have implicit meaning, it can be called a database.

Database Management System: A database management system (DBMS) is a collection of programs that enables users to create and maintain a database.
The DBMS is a general-purpose software system for

- Defining databases
- Constructing databases
- Manipulating databases and
- Sharing databases among various users and applications.

Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database.
Manipulating a database includes functions such as to retrieve specific data (query), updating the database, etc.
Sharing a database allows multiple users and programs to access the database simultaneously.
Other important functions provided by the DBMS include protecting the database and maintaining it over a long period of time.
DBMS Examples : Oracle, MySQL, Sql Server, DB2, Ingress, . . .

Example: University Database

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

This University database stores student and course information.The database is a collection of tables. A table has a set of named columns. Each column is of some data type. Database *manipulation* involves *querying* and *updating*.
Example query: Retrieve class of 'Smith'.
Example of update: Change the class of ‘Smith’.

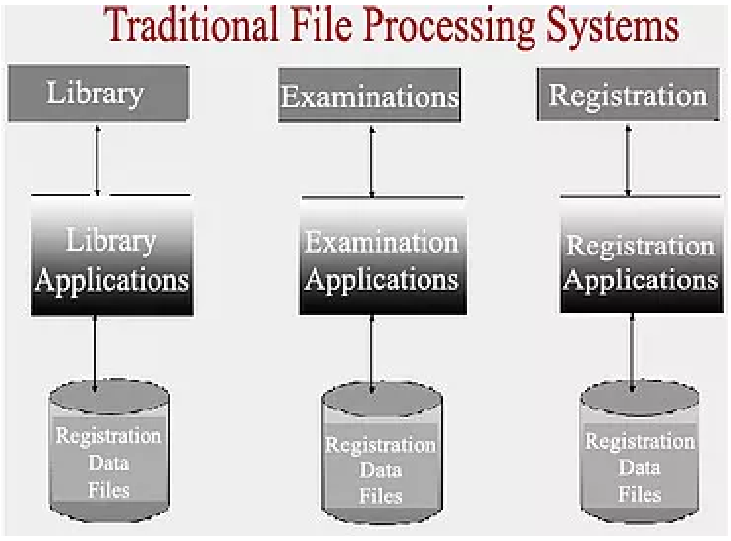
Some database applications are

Universities: For student information, registration, and grades
Banking: For customer information, accounts, loans and banking transactions
Airlines: For reservations, and schedule information
Sales: For customer, product, purchase information
Manufacturing: For production, inventory, orders, and supply chain
Human resources: For employee records, salaries, and tax deductions

Difference Between File Processing System and Database Approach:

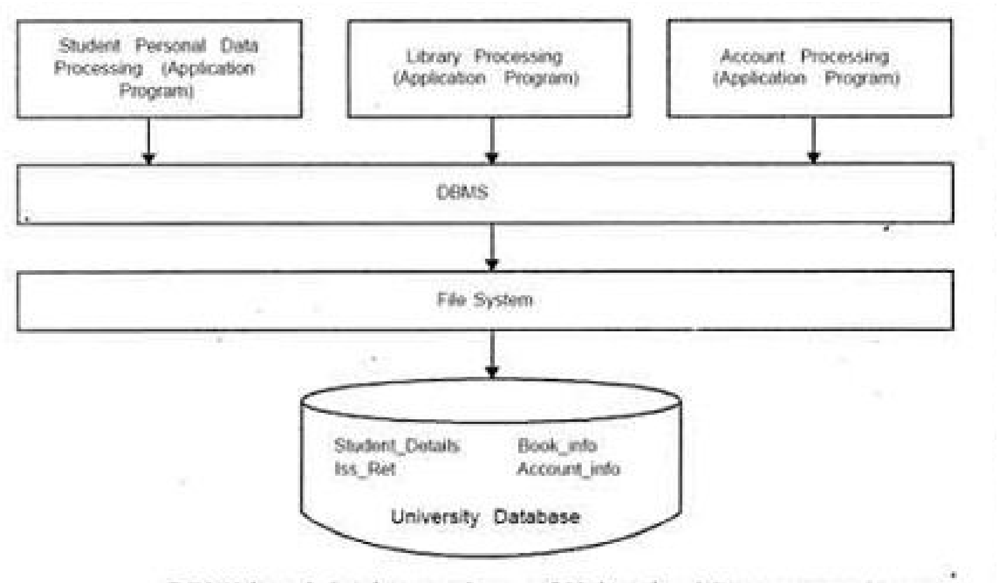
1. File Processing System:

In the traditional file-based approach, each department in an organization has its own set of separate files and the records. The data in one file may not relate to the data in any other file.



Database Approach:

When an organization uses the database approach, it allows many programs and users to share the data in the database. Authorized users can access, retrieve and select data by a single query from the database. As shown in the below image, various areas within the university share and interact with the data in the database.


Student Personal Data Processing (Application Program)
Library Processing (Application Program)
Account Processing (Application Program)
DBMS
File System
Student_Details
Iss_Ret
Book_info
Account_info
University Database

DBMS based implementation of University Management System

In the **database approach**, a single database is maintained that is used by all the departments in the organization.

Disadvantages of Traditional File processing:

1. Data Redundancy – Data redundancy occurs when the same data items are stored in multiple files. Each department in an organization has its own separate files in the file processing system. For example, the Student contract file and the Student file store the same students' names and addresses.

2. **Wastes resources** - Duplicating data in this way is a waste of storage space. People's time is also wasted because file maintenance tasks require more time in updating or deleting multiple files that contain the same data whenever the data is modified. It also Increases the chance of errors. For example, if a student changes his or her telephone numbers, the institution must update the student contact file. If no change is made in all the files where the data is stored, then **inconsistencies** among the files exist.

3. **Isolated Data** – Isolated data increases the difficulty in accessing the data that is stored in separate files in different departments. **Sharing** data from multiple and separate files is a complicated procedure especially when there are many files involved.

Characteristics of the Database Approach

The main characteristics of the database approach versus the file-processing approach are the following:

1. Self-describing nature of a database system
2. Insulation between programs and data, and data abstraction
3. Support of multiple views of the data
4. Sharing of data and multi user transaction processing

1. Self-Describing Nature of a Database System:

A database system includes

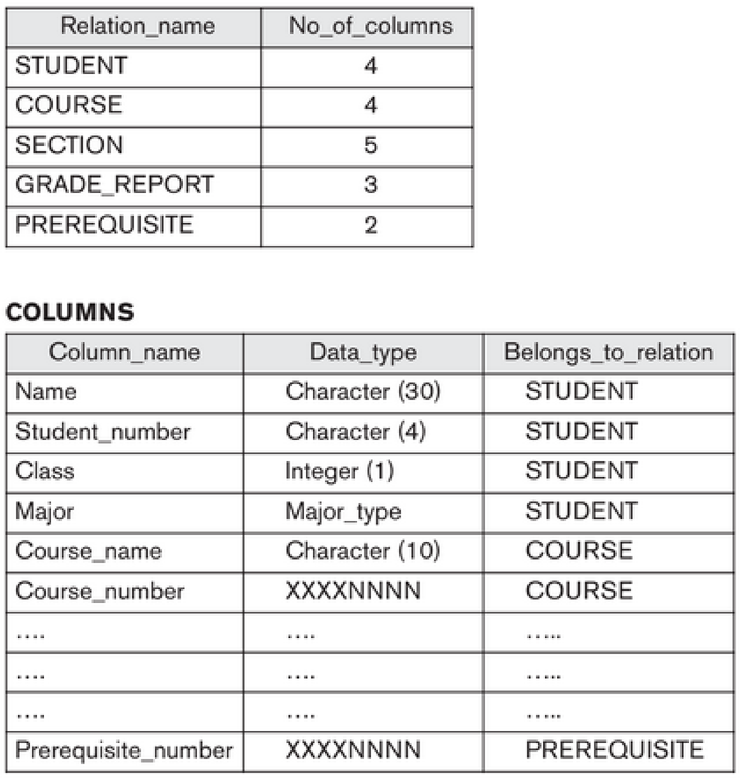
- the database
- the complete definition of the database structure and
- constraints (limitations).

The information such as the structure of each table, the type and storage format of each data item, and various constraints on the data are stored in the **DBMS catalog**, which is called **meta-data** (data about data)**.** It is also called a **data dictionary.**

The catalog is used by the DBMS software and also by database users who need information about the database structure. These definitions are specified by the database designer before creating the actual database and are stored in the catalog.

For example whenever a request is made to access the Name of a STUDENT record, the DBMS software refers to the catalog to determine the structure of the STUDENT table and the position and size of the Name data item within a STUDENT record. A part of catalog for the University database is given below:

TABLES



Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	.....
....	....	.....
....	....	.....
Prerequisite_number	XXXXNNNN	PREREQUISITE

2. **Insulation between Programs and Data; Data Abstraction:**
The structure of data files is stored in the DBMS catalog separately from the access programs.
Program-Data Independence: In traditional file processing, the structure of the data files accessed by an application is "hard-coded" in its source code. If we want to change the structure of the data (e.g., by adding the first two digits to the YEAR field), every application in which a description of that file's structure is hard-coded must be changed.
But the DBMS access programs do not require such changes, because the structure of the data is described in the system catalog, separately from the programs.
In other words, the DBMS provides a conceptual or logical view of the data to application programs, so that the underlying implementation may be changed without the programs being modified. This is referred to as ***program-data independence***.
Also, which access paths (e.g., **indexes**) exist are listed in the catalog, helping the DBMS to determine the most efficient way to search for items in response to a query.

3. Multiple Views of Data (Data Abstraction): Different users (e.g., in different departments of an organization) have different "views" on the database. A good DBMS has facilities for defining multiple views. This is not only convenient for users, but also **provides security** in data access.

For example, from the point of view of an institution's office employee, student data does not include which grades were earned. As another example, a university's office employee might think that GPA is a field of data in each student's record. In reality, the underlying database might calculate that value each time it is needed. This is called **virtual** (or **derived**) data.

4. Data Sharing and Multi-user Transaction Processing: A multi-user DBMS must allow multiple users to access the database at the same time. The DBMS must include concurrency control software.

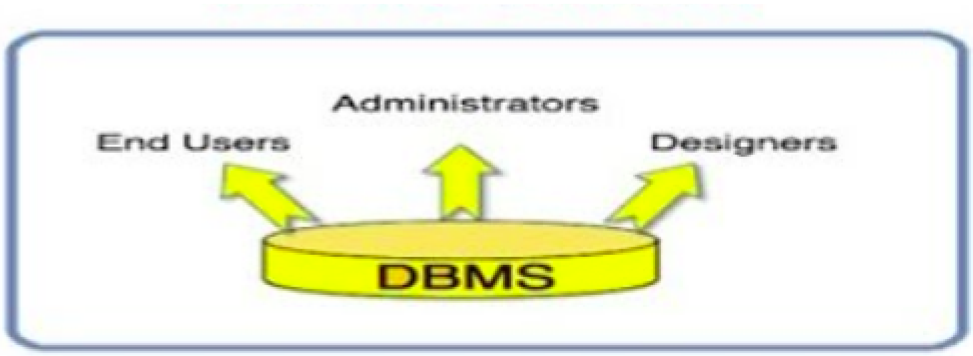
The concurrency control Manager of the DBMS ensures that several users can update the same data in a "controlled" manner. A fundamental role of multi user DBMS software is to ensure that concurrent transactions operate correctly and efficiently.

A **transaction** is an executing program or process that includes one or more database accesses, such as reading or updating of database records. Each transaction should execute a logically correct database access without interference from other transactions. The DBMS must enforce several transaction properties.
i) The **atomicity property** ensures that either all the database operations in a transaction are executed or none are.
ii) The **isolation property** ensures that each transaction appears to execute in isolation from other transactions, even though hundreds of transactions may be executing concurrently.

Applications such as **airline reservation systems** are known as **online transaction processing** applications.

Database Users

Many people are involved in the design, use, and maintenance of a large database with hundreds of users.



1. Database Administrators (DBA)

A person who has overall control over the database is called DBA. Administering the database resources is the responsibility of the database administrator (DBA).
The DBA is responsible for
- i) authorizing users to access the database
- ii) coordinating / monitoring its use
- iii) acquiring hardware/software for upgrades, etc.

In large organizations, the DBA might have a supporting staff.

2. Database Designers

Database designers are responsible for
- i) **identify** the **data to be stored** in the database
- ii) **choose** appropriate **structures** to represent and **store this data**.
- iii) **communicate** with all **database users** to understand their requirements and
- iv) **create a design** that meets these requirements.

3. End Users

End users are the people whose jobs require access to the database for querying,

updating, and generating reports. The database primarily exists for their use.
There are several categories of end users:

■ **Casual end users** use the database occasionally. But they may need different information each time. They use query language to specify their requests. Casual users learn only a few facilities that they may use repeatedly.

■ **Naive or parametric end users** make up a major portion of database end users. (Naive means no experience. Parametric End Users have no DBMS knowledge but they frequently use the database applications in their daily life to get the desired results). They **use program interfaces** to use the database system. Eg: Bank tellers, Reservation agents for airlines, hotels car rental companies etc.

■ **Sophisticated end users** include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS. (They have knowledge about the database). They implement their own applications to meet their complex requirements. Sophisticated users try to learn most of the DBMS facilities in order to achieve their complex requirements.

■ **Standalone users** maintain personal databases. They use ready made program packages that provide easy-to-use menu-based or graphics-based interfaces. Standalone users are experts in using a specific software package.

4. System Analysts and Application Programmers
System analysts determine the requirements of end users, especially naive and parametric end users. They **develop specifications** for standard transactions that meet these requirements.
Application programmers implement these specifications as **programs**. Such **analysts** and **programmer**s should be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks.

<u>Advantages of Using the DBMS Approach</u>

The 4 main characteristics of dbms are:

1. Self-Describing Nature of a Database System
2. Insulation between Programs and Data; Data Abstraction
3. Multiple Views of Data
4. Data Sharing and Multi-user Transaction Processing

In addition dbms provide some additional capabilities. They can be listed as follows:

1. Controlling Redundancy
Redundancy in storing the same data multiple times leads to several problems. They are:

i) **Duplication of Effort :** There is the need to update multiple copies of the same data item.
ii) **Storage space is wasted** when the same data is stored repeatedly.
iii) Files that represent the same data may become **inconsistent**.

A DBMS should provide the capability that no inconsistencies are introduced when data is updated. **Data normalization** ensures consistency and saves storage space.

2. Restricting Unauthorized Access
When multiple users share a large database, most users will not be authorized to access all information in the database. A DBMS should provide a **security and authorization subsystem**, which is used for specifying restrictions on user accounts

3. Providing Persistent Storage for Program Objects: Object-oriented database systems make it easier for complex runtime objects (e.g., lists, trees) to be saved in secondary storage so as to survive beyond program termination and to be retrievable at a later time.

4. Providing Storage Structures and Search Techniques for Efficient Query Processing :
The DBMS must provide specialized data structures and search techniques like **indexes** to speed up disk search for the desired records. The **query processing and optimization** module is responsible for choosing an efficient query execution plan for each query submitted to the system.
(Query means request for information.)

5. Providing Backup and Recovery
A DBMS must provide facilities for recovering from hardware or software failures.The backup and recovery subsystem of the DBMS is responsible for recovery.

6. Providing Multiple User Interfaces:
Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces.
For example, query languages for casual users, programming language interfaces for application programmers, forms and/or command codes for parametric users, menu-driven interfaces for stand-alone users.

7. Representing Complex Relationships among Data:
A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.

8. Enforcing Integrity Constraints
Most database applications require certain restrictions to be held for the data. A DBMS should provide capabilities for defining and enforcing these constraints. (E.g : A Grade field can include only the values in the set { "A", "A-", "B+", ..., "F" }.

9. Permitting Inferencing and Actions Using Rules
Some database systems provide capabilities for defining **deduction rules** for getting new information from the stored database facts. Such systems are called deductive database systems.

Additional Implications of Using the Database Approach .

1. Potential for Enforcing Standards.

The database approach permits the DBA to define and enforce standards among database users in a large organization. (use of national standard codes pr international standard codes)

2. **Reduced Application Development Time**.

Once a database is up and running, less time is required to create new applications using DBMS facilities.

3. Flexibility.

We can change the structure of a database as requirements change, without affecting the stored data and the existing application programs.

4. Availability of Up-to-Date Information

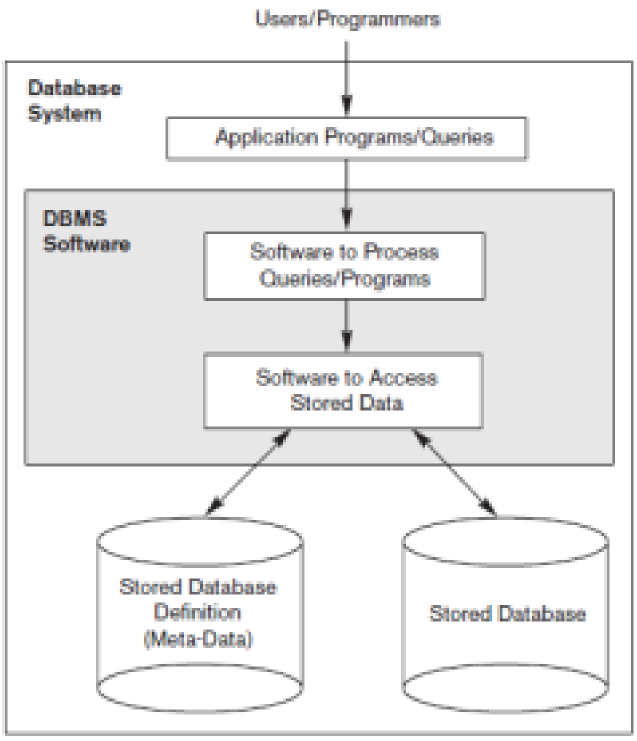
As soon as one user's update is applied to the database, all other users can immediately see this update.

5. Economies of Scale.

DBMS enables the whole organization to invest in more powerful processors, storage devices, or communication gear, rather than having each department purchase its own (lower performance) equipment. This reduces overall costs of operation and management.

6. Improved Data Integrity – When users modify data in the database, they can directly make changes to one file instead of multiple files. Therefore, the mismatch in different copies of the same data will not occur. This database approach increases the quality and data's integrity by reducing the possibility of causing inconsistencies.

Database System: The database and DBMS software together is called a database system. Figure shows a simplified database system environment.


Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Data Models

Data abstraction generally refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data.
A **data model** is a collection of concepts that can be used to describe the structure of a database. It provides the necessary means to achieve data abstraction. By structure of a database we mean the data types, relationships, and constraints that apply to the data. Most data models also include a set of basic operations for specifying retrievals and updates on the database.

Categories of Data Models
Data models can be classified into 3 categories.
1. Low-level Data Model / Physical Data Model.
2. Representational DM / Implementation DM / Record based model
3. High-level Data Model / conceptual Data Model

1. Low-level data models provide concepts that describe the details of how data is stored in the computer. (in magnetic disks). Low-level data models are also called physical data models. Concepts provided by low-level data models are generally meant for computer specialists, not for end users. (Physical Model)

2. Representational (or implementation) data models provide concepts that may be easily understood by end users. 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. Representational data models represent data by using record structures and hence are called **record-based data models.** Examples: widely used **relational data model**, as well as the **network data model and hierarchical data models** that have been widely used in the past.

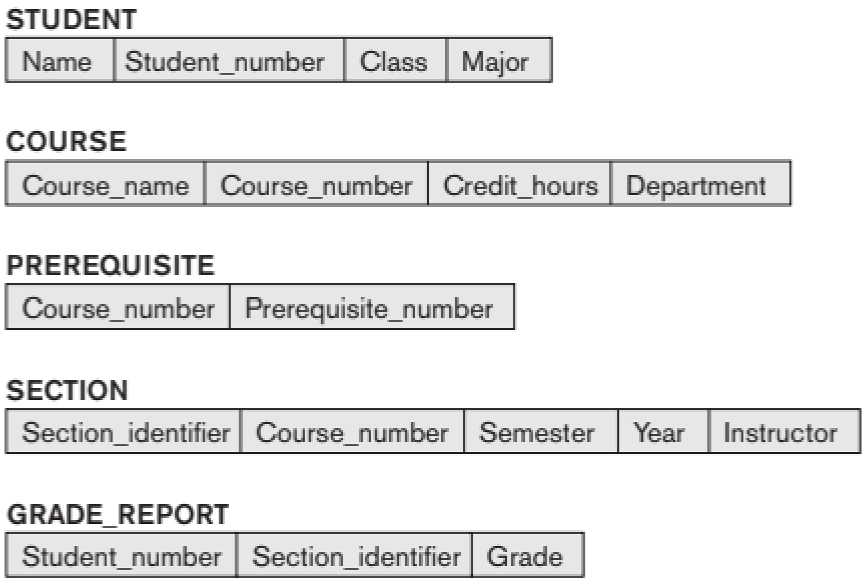
3. High-level or conceptual data models provide concepts that are close to the way many users visualize data. Example: **Entity-Relationship model** is a popular high-level conceptual data model. The object data model is an example of a new family of higher-level implementation data models that are closer to conceptual data models.

Schemas & Instances

Schema: The description of a database is called the **database schema**. It is specified during database design and is not expected to change frequently.

A displayed schema is called a **schema diagram**. A schema diagram only shows us the database design. It does not show the actual data of the database. A schema diagram is a diagram which contains entities and the attributes that will define that schema.

The diagram displays the structure of each record type. The following Figure shows a schema diagram for the University database.



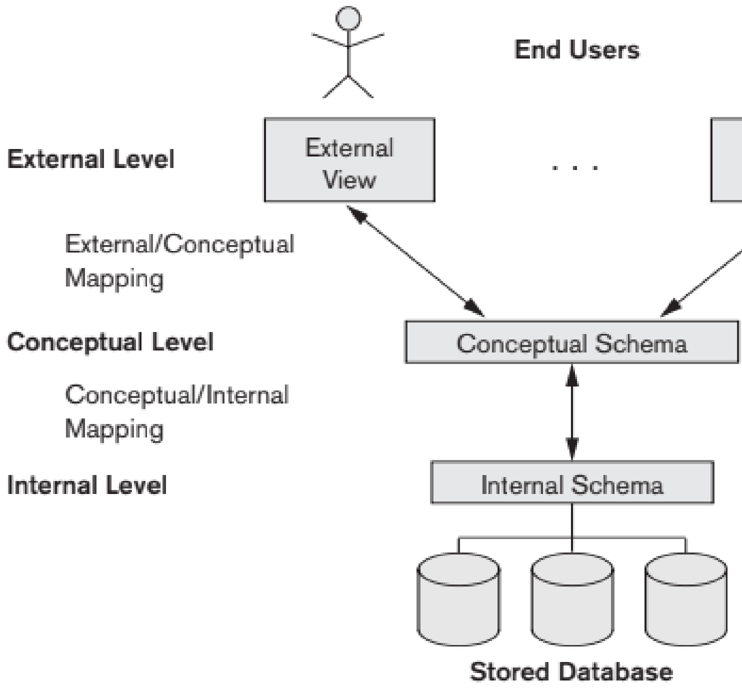
A schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints. Each object in the schema is called **schema construct**. (such as STUDENT or COURSE)

Instance: The data in the database at a particular moment in time is called a **database instance** or database state or snapshot. It is also called the current set of occurrences in the database.

Every time we insert or delete a record or change the value of a data item in a record, we change one state of the database into another state. In a given database state, each schema construct has its own current set of instances. For example, the STUDENT construct will contain the set of individual student entities (records) as its instances.

The schema is sometimes called the **intension**, and a database state is called an **extension of the schema**.

Three-Schema Architecture and Data Independence



In the three schema architecture, schemas can be defined at the following three levels:

1. **The internal level or physical level** has an **internal schema**, which 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**, which describes the structure of the whole database. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
3. **The external level 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.

The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system. In a DBMS based on the three-schema architecture, each user group refers to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database.

Mapping is used to transform the request and response between various database levels of architecture. In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema. In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

Data Independence:

Data independence is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. There are 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. The conceptual schema may be expanded (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).After the conceptual schema undergoes a logical reorganization, application programs that reference the external schema constructs must work as before. Changes to constraints can be applied to the conceptual schema without affecting the external schemas or application programs.

2. **Physical data independence** is the capacity to change the internal schema without changing 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.

Physical data independence exists in most databases and file environments where physical details such as the exact location of data on disk, and hardware details of storage encoding, placement, compression, splitting, merging of records, and so on are hidden from the user. Applications can remain unaware of these details.

On the other hand, logical data independence is harder to achieve because it allows structural and constraint changes without affecting application programs—**a much stricter requirement**.

Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged. Only the mapping between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.

Database Languages

The DBMS must provide appropriate languages and interfaces for each category of users. In the database world, the two functions of declaration and manipulation are separated into two different languages DDL and DML.

DDL: In many DBMSs, **the data definition language (DDL**), is used by the DBA and by database designers to **define schemas**. The DBMS has 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.

The **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.

A **view definition language (VDL)** is used 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.

Examples of DDL commands in SQL are CREATE TABLE, ALTER TABLE, DROP TABLE, and TRUNCATE TABLE.

DML: The database manipulation includes 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 **non-procedural DML (or high-level DML)** can be used to specify complex database operations in a short and clear way (concisely).

A **procedural DML (or low-level 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.

Examples of DML commands in SQL are SELECT, UPDATE, DELETE, and INSERT.

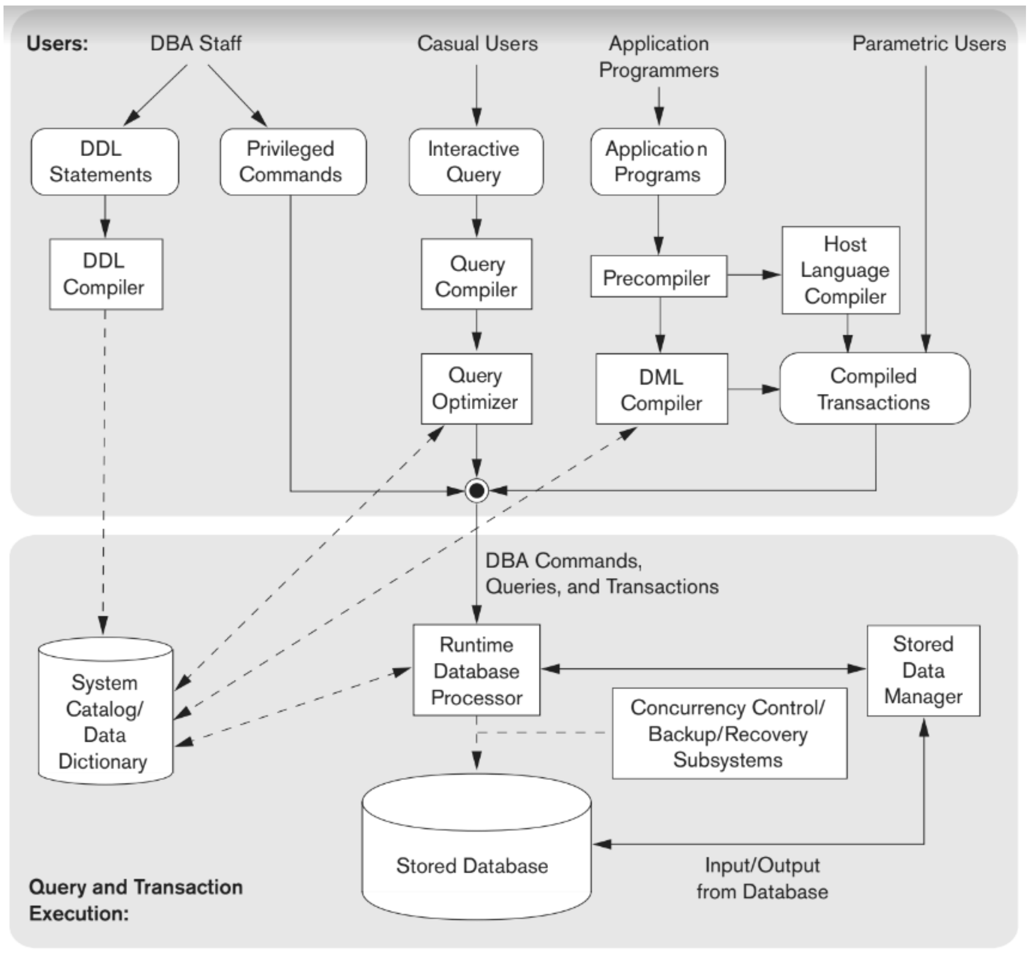
The Database System Environment
(Database System Architecture)

A DBMS is a complex software system. The database system environment includes the types of software components that constitute a DBMS and the types of system software with which the DBMS interacts.

DBMS Component Modules:

Figure shows the DBMS components. The figure is divided into two parts.

1. The top part of the figure shows the various users of the database environment and their interfaces.
2. The lower part shows the internal components of the DBMS which are responsible for storage of data and processing of transactions.



The top part of Figure shows the following users and their intefaces

i) **DBA staff**

ii) **casual users**

iii) **application programmers**

iv) **parametric users** 3

i) The DBA staff works on defining the database and making changes to the database definition. They use the DDL and other privileged commands.

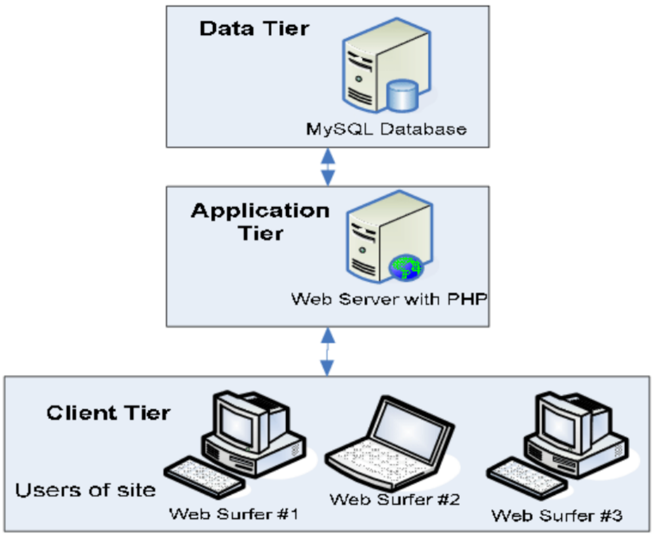
The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.
The catalog includes information such as the names of files, sizes of files, data item names and data types of data items, storage details of each file, mapping information among schemas, and constraints. (In addition, the catalog stores many other types of information that are needed by the DBMS modules.)
ii) Casual users occasionally need information from the database. They use interfaces to formulate queries. These queries are parsed and validated for correctness of the query syntax by a **query compiler.** The **query compiler** compiles them into an internal form. (This internal query is subjected to query optimization.) The **query optimizer** optimizes the query by the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes.
iii) **Application programmers** write programs in host languages such as Java, C, or C++. They are submitted to a precompiler. The **precompiler** extracts DML commands from the application program. These commands are sent to the DML compiler for compilation into object code for database access. The rest of the program is sent to the host language compiler. The object codes for the DML commands and the rest of the program are linked, forming a canned transaction (standard transaction).
iv) Canned transactions are executed repeatedly by **parametric users**, who simply supply the parameters to the transactions. Each execution is considered to be a separate transaction. An example is a bank withdrawal transaction.

In the lower part, the **runtime database processor** executes (1) the privileged commands, (2) the executable query plans, and (3) the canned transactions with runtime parameters. It works with the **system catalog**. It also works with the **stored data manager**, which uses basic operating system services. The runtime database processor handles other aspects of data transfer, such as management of buffers in the main memory. Some DBMSs have their own buffer management module while others depend on the OS for buffer management.

Concurrency control and **backup and recovery systems** are integrated into the working of the runtime database processor.

Usually we have a **client program** that accesses the DBMS running on a separate computer from the computer on which the database resides. The former is called the **client computer** running a DBMS client software and the latter is called the **database server**. In some cases, the client accesses a middle computer, called the **application server**, which in turn accesses the database server.

(Figure: Three tier architecture:



)

The DBMS interacts with the operating system when disk accesses are needed. If the computer system is mainly dedicated to running the database server, the DBMS will control the main memory buffering of disk pages. The DBMS also interfaces with compilers for general purpose host programming languages, and with application servers and client programs running on separate machines through the system network interface.