Database Management Systems

MODULE 1

Introduction & Entity Relationship (ER) Model

Athira S. Nair, AP, CS, MEC

.

Concept & Overview of Database Management Systems (DBMS)

- A **database** is a collection of related data.
- **Data** means known facts that can be recorded and that have implicit meaning
- A database has the following implicit properties:
    - A database represents some aspect of the real world, sometimes called the **miniworld** or the **universe of discourse (UoD)**. Changes to the miniworld are reflected in the database.
    - A database is a logically coherent collection of data with some inherent meaning.
    - A random assortment of data cannot correctly be referred to as a database.
    - A database is designed, built, and populated with data for a specific purpose.

It has an intended group of users and some preconceived applications in which these users are interested

In essence, a database has:

- **Source of data.**
- **Interaction with real world.**
- **Audience interested in that data.**

Database Management Systems

- 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*** that facilitates the processes of ***defining, constructing, manipulating,*** **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.
- The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called **meta-data**.
- **Constructing** the database is the process of storing the data on some storage medium that is controlled by the DBMS.
- **Manipulating** a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.
- **Sharing** a database allows multiple users and programs to access the database simultaneously.

- An **application program** accesses the database by sending queries or requests for data to the DBMS.
- A **query** typically causes some data to be retrieved; a transaction may cause some data to be read and some data to be written into the database.
- Other important functions provided by the DBMS include protecting the database and maintaining it over a long period of time.
- **Protection** includes system protection against hardware or software malfunction (or crashes) and security protection against unauthorized or malicious access.
- A typical large database may have a life cycle of many years, so the DBMS must be able to maintain the database system by allowing the system to evolve as requirements change over time.
- To complete our initial definitions, we will call the database and DBMS software together a **database system**.

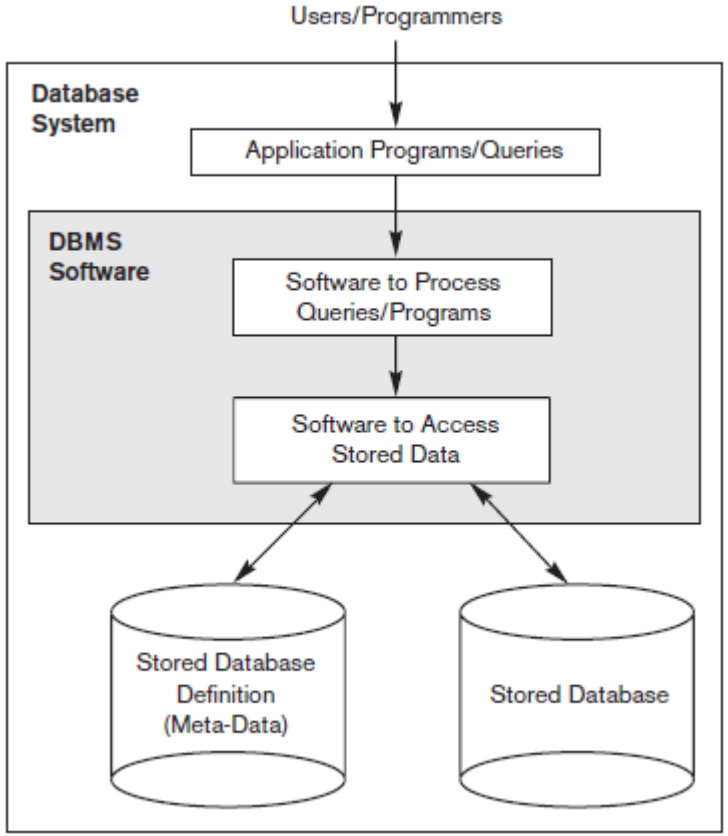


Figure 1.1
A simplified database system environment.

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

Figure 1.2
A database that stores student and course information.

Characteristics of the Database Approach

- In traditional file processing, each user defines and implements the files needed for a specific software application as part of programming the application.
- There is redundancy in defining and storing data which results in wasted storage space and in redundant efforts to maintain common data up to date.
- In the database approach, a single repository of data is maintained that is defined once and then is accessed by various users.
- In file systems, each application is free to name data elements independently.
- In contrast, in a database, the names or labels of data are defined once, and used repeatedly by queries, transactions, and applications.
- Main characteristics of the database approach versus the file-processing approach

    - **Self-describing nature of a database system**
    - **Insulation between programs and data, and data abstraction**
    - **Support of multiple views of the data**
    - **Sharing of data and multiuser transaction processing**

1. Self-Describing Nature of a Database System

- The database system contains not only the database itself but also a complete definition or description of the database structure and constraints.
- This definition is stored in the **DBMS catalog**, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data.
- The information stored in the catalog is called **meta-data**, and it describes the structure of the primary database.
- A general-purpose DBMS software package is not written for a specific database application, and hence it must refer to the catalog to know the structure of the files in a specific database, such as the type and format of data it will access.
- In traditional file processing, data definition is typically part of the application programs themselves.
- Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs.

2. Insulation between programs and data, and data abstraction

- In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access this file.
- By contrast, DBMS access programs do not require such changes in most cases.
- The structure of data files is stored in the DBMS catalog separately from the access programs.
- This is called property **program-data independence**
- In some types of database systems, such as object-oriented and object-relational systems, users can define operations on data as part of the database definitions.
- An **operation** (also called a *function* or *method*) is specified in two parts.
- The *interface* (or *signature*) of an operation includes the operation name and the data types of its arguments (or parameters).
- The *implementation* (or *method*) of the operation is specified separately and can be changed without affecting the interface.
- User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented.
- This may be termed **program-operation independence**

- The characteristic that allows program-data independence and program-operation independence is called **data abstraction**.
- A DBMS provides users with a **conceptual representation** of data that does not include many of the details of how the data is stored or how the operations are implemented.
- Informally, a **data model** is a type of data abstraction that is used to provide this conceptual representation.
- The data model uses logical concepts, such as objects, their properties, and their interrelationships, that may be easier for most users to understand than computer storage concepts.
- Hence, the data model *hides* storage and implementation details that are not of interest to most database users.

3. Support of Multiple Views of the Data

- A database typically has many users, each of whom may require a different perspective or view of the database.
- A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.
- Some users may not need to be aware of whether the data they refer to is stored or derived.
- A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views

(a)

TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

(b)

COURSE_PREREQUISITES

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

Figure 1.5
Two views derived from the database in Figure 1.2. (a) The TRANSCRIPT view. (b) The COURSE_PREREQUISITES view.

4. Sharing of Data and Multiuser Transaction Processing

- A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time.
- This is essential if data for multiple applications is to be integrated and maintained in a single database.
- The DBMS must include **concurrency control** software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.
- 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 is supposed to execute a logically correct database access if executed in its entirety without interference from other transactions.

Database Users

“**Actors on the scene**" are those who use and control a database while "**workers behind the scene**" are those who design, develop, and maintain the database system environment

Actors on the scene

These are the people who use a database on a daily basis, such as

- database administrators,
- database designers,
- end users,
- system analysts, and
- application programmers

1. Database administrators:

- In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software.
- Administering these resources is the responsibility of the database administrator (DBA).
- The DBA is responsible for authorizing access to the database, for coordinating and monitoring its use, and for acquiring software and hardware resources as needed.
- The DBA is accountable for problems such as breach of security or poor system response time.

2. Database Designers

- Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data.
- It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements, and to come up with 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** occasionally access the database, but they may need different information each time. They are typically middle-or high-level managers or other occasional browsers

  - **Naive or parametric end users**
  - Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates-called ***canned transactions***-that have been carefully programmed and tested. Bank tellers, Reservation Clerks for airlines, hotels, etc. are the examples

  - **Sophisticated end users** include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS so as to implement their applications to meet their complex requirements.

  - **Stand-alone users** maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces.

4. System analysts and application programmers (Software Engineers)

● **System analysts *determine the requirements*** of end users, especially naive and parametric end users, and ***develop specifications*** for canned transactions that meet these requirements.

● **Application programmers *implement these specifications*** as programs; then they test, debug, document, and maintain these canned transactions

- Such analysts and programmers—commonly referred to as **software developers** or **software engineers**—should be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks.

WORKERS BEHIND THE SCENE

- Some users are associated with the design, development, and operation of the DBMS software and system environment. They are:

● **DBMS system designers and implementers** are persons who design and implement the DBMS modules and interfaces as a software package.

● **Tool developers** design and implement **tools**—the software packages that facilitate database modeling and design, database system design, and improved performance.

- Tools are optional packages that are often purchased separately. They include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulation, and test data generation..

● **Operators and maintenance personnel** are the system administration personnel who are responsible for the actual running and maintenance of the hardware and software environment for the database system

What are unstructured, structured, and semi-structured data?

1. Structured Data

- **Definition:** Highly organized data that resides in a fixed format, such as tables with rows and columns.

- **Characteristics:**
  - Easy to store, access, and analyze using relational databases and tools like SQL.
  - Has a predefined schema (e.g., data types, relationships).
  - Ideal for well-defined and predictable use cases.

- **Examples:**
  - Relational databases (e.g., MySQL, PostgreSQL).
  - Spreadsheets.
  - Financial transactions.
- **Usage:** Banking systems, inventory management, customer databases.

2. Unstructured Data

- **Definition:** Data without a predefined format or organization. It's often stored in its raw form and requires processing to extract meaningful insights.
- **Characteristics:**
  - Difficult to analyze directly without specialized tools.
  - Can include text, images, audio, and video.
  - No fixed schema.
- **Examples:**
  - Social media posts.
  - Emails.
  - Images, audio, and video files.
  - IoT sensor data.
- **Usage:** Social media analysis, multimedia storage, and NLP tasks.

3. Semi-structured Data

- **Definition:** Data that does not fit into a strict relational structure but still has some organizational properties, such as tags or markers, that make it easier to analyze than unstructured data.
- **Characteristics:**
  - Has elements of both structured and unstructured data.
  - Often uses formats that include metadata (descriptive tags).
  - Can be stored in document-oriented databases or NoSQL databases.
- **Examples:**
  - JSON and XML files.
  - Log files.
  - Sensor data with labeled fields.
  - Email with metadata (sender, recipient, timestamp) but free-text content.
- **Usage:** Data exchange between systems, APIs, and IoT applications.

Structured

Data is well organized.

Organized by the means of relational databases.

Matured transaction, multiple concurrency techniques.

Tuples, rows and tables.

Schema dependent and less flexible.

Query performance is the highest, structured query can be performed allowing complex joins.

Semi-structured

Data is organized to some extent.

Partially organized, e.g. by XML/RDF.

Transaction is adapted from DBMS, but data concurrency can pose problems.

Tuples or graphs are possible.

Data is more flexible than structured.

Queries over anonymous nodes are possible.

Unstructured

Data is fully non organized.

Based on character and binary data.

Difficult, but achievable transaction management and data concurrency.

Versioning usually on whole data or chunks.

The most flexible.

Schema on-read so query performance is the lowest.

ADVANTAGES OF DBMS

1. Controlling Redundancy:

Redundancy in storing the same data multiple times leads to several problems such as:

- *Duplication of effort.*
- *Storage space* is *wasted*
- Files that represent the same data may become *inconsistent.*

In the database approach, we should have a database design that stores each logical data item—such as a student's name or birth date—in *only one place* in the database. This is known as **data normalization**, and it ensures consistency and saves storage space
In DBMS, all the files are integrated in a single database. So there is no chance of duplicate data.
Thus in practice, it is sometimes necessary to use **controlled redundancy** for improving the performance of queries.

2. Restricted Unauthorized Access

- A DBMS should provide a **security and authorization subsystem**, which the DBA uses to create accounts and to specify account restrictions.
- Then, the DBMS should enforce these restrictions automatically.
- Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database.
- Data security level is high by protecting your precious data from unauthorized access. Only authorized users should have the grant to access the database with the help of credentials.

3. Providing Persistent Storage for Program Objects

- Databases can be used to provide persistent storage for program objects and data structures.
- This is one of the main reasons for object-oriented database systems.
- The values of program variables are discarded once a program terminates, unless the programmer explicitly stores them in permanent files,
- The persistent storage of program objects and data structures is an important function of database systems.
- Traditional database systems often suffered from the so called **impedance mismatch problem**, since the data structures provided by the DBMS were incompatible with the programming language's data structures.

4. Providing Storage Structures for Efficient Query Processing

- Database systems must provide capabilities for efficiently executing queries and updates.
- Because the database is typically stored on disk, the DBMS must provide specialized data structures to speed up disk search for the desired records.
- Auxiliary files called **indexes** are used for this purpose.
- Indexes are typically based on tree data structures or hash data structures, suitably modified for disk search.
- In order to process the database records needed by a particular query, those records must be copied from disk to memory.
- Hence, the DBMS often has a **buffering** module that maintains parts of the database in main memory buffers.
- The **query processing and optimization** module of the DBMS is responsible for choosing an efficient query execution plan for each query based on the existing storage structures

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.
- For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.

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.
- These include query languages for casual users, programming language interfaces for application programmers, forms and command codes for parametric users, and menu-driven interfaces and natural language 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 as well as to retrieve and update related data easily and efficiently.

8. Enforcing Integrity Constraints

- Most database applications have certain integrity constraints that must hold for the data.
- A DBMS should provide capabilities for defining and enforcing these constraints.
- The simplest type of integrity constraint involves specifying a data type for each data item.
- These constraints are derived from the meaning or semantics of the data and of the miniworld it represents.
- It is the database designers' responsibility to identify integrity constraints during database design.
- Some constraints can be specified to the DBMS and automatically enforced.
- Other constraints may have to be checked by update programs or at the time of data entry.

9. Permitting Inferencing and Actions Using rules

- Some database systems provide capabilities for defining *deduction rules* for inferencing new information from the stored database facts.
- Such systems are called deductive database systems.

10. Additional Implications of Using the Database Approach

- **Potential for Enforcing Standards:** The database approach permits the DBA to define and enforce standards among database users in a large organization.
- **Reduced Application Development Time:** A prime selling feature of the database approach is that developing a new application-such as the retrieval of certain data from the database for printing a new report-takes very little time.
- **Flexibility:** It may be necessary to change the structure of a database as requirements change.
- **Availability of Up-to-Date Information:** A DBMS makes the database available to all users.
- **Economies of Scale**: The DBMS approach permits consolidation of data and applications, thus reducing the amount of wasteful overlap between activities of data processing personnel in different projects or departments

Data Abstraction: Simplifying Complexity

Data abstraction is a key feature of the database approach, allowing users to perceive data at their desired their desired level of detail.

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

Data models provide the means to achieve data abstraction

Data Models

- **Data models** provide the means to achieve data abstraction by defining the structure of a database—data types, relationships, and constraints—along with operations for retrieving and updating data.
- Data models describe how a database's logical structure is represented. In a database management system, data models are essential for introducing abstraction.
- Data models specify how data is linked to one another, as well as how it is handled and stored within the system.

Conceptual

- Close to the way users perceive data.
- Also called high level

Physical

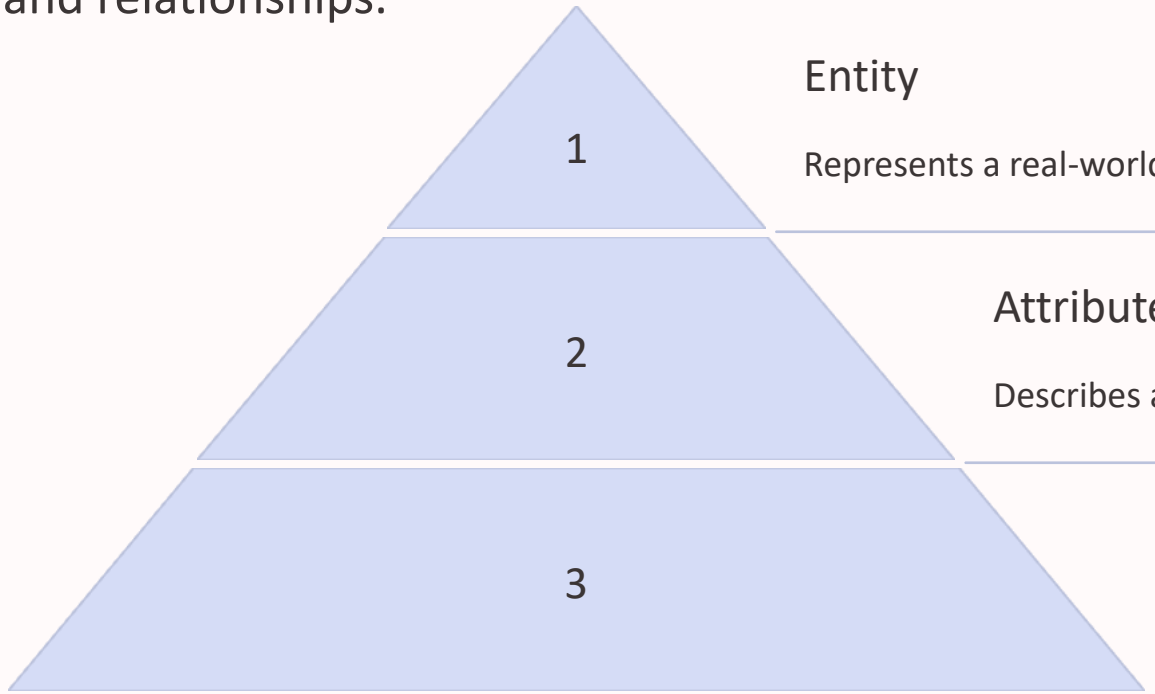
- Details of data storage on computer media.
- Also called low level model

Representational

- Bridge between user understanding and computer computer storage.
- Also called implementation data data model

Conceptual Data Models: Entities, Attributes, and Relationships

Conceptual models represent data in a way that is close to how users perceive it, focusing on real-world on real-world objects and their properties. They employ concepts such as entities, attributes, and and relationships.



Entity

Represents a real-world object or concept.

Attribute

Describes a property of an entity.

Relationship

Association between entities.

Representational Data Models

- This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database.
- The representational data model allows us to focus primarily, on the design part of the database.
- A popular representational model is a Relational model.
- The relational Model consists of Relational Algebra and Relational Calculus.
- In the Relational Model, we basically use tables to represent our data and the relationships between them.
- It is a theoretical concept whose practical implementation is done in Physical Data Model.
- The advantage of using a Representational data model is to provide a foundation to form the base for the Physical model

1 Relational Model

Organizes data into tables with rows rows and columns.

2 Network Model

Connects data using a network-like structure.

3 Hierarchical Model

Organizes data in a tree-like structure.

Physical Data Models: Storage and Access

- Physical models describe how data is stored as files within a computer, including details like record formats, orderings, and access paths.
- The physical Data Model is used to practically implement Relational Data Model.
- Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes.
- This is stored in the form of files, records, and certain other data structures.
- It has all the information on the format in which the files are present and the structure of the databases, the presence of external data structures, and their relation to each other.
- Here, we basically save tables in memory so they can be accessed efficiently.
- In order to come up with a good physical model, we have to work on the relational model in a better way.
- This Data Model describes **HOW** the system will be implemented using a specific DBMS. This model is typically created by DBA and developers. The purpose is actual implementation of the database.
- These concepts are primarily for computer specialists.
- An **access path** is a structure that makes the search for particular database records efficient.
- An index, for instance, allows direct access to data using an index term or keyword, similar to a book index.

Schemas, Instances, and Database State

- In any data model, it's crucial to distinguish between the description of the description of the database (schema) and the actual data itself (database (database state or snapshot).
- The description of a database is called the **database schema**, which 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**.
- Each object in the schema—such as STUDENT or COURSE—a **schema construct**.
- The actual data in a database may change quite frequently.
- The data in the database at a particular moment in time is called a **database state** or **snapshot or instance**.
- It is also called the *current* set of **occurrences i**n the database
- As data is updated (inserted, deleted, modified), the database state changes.

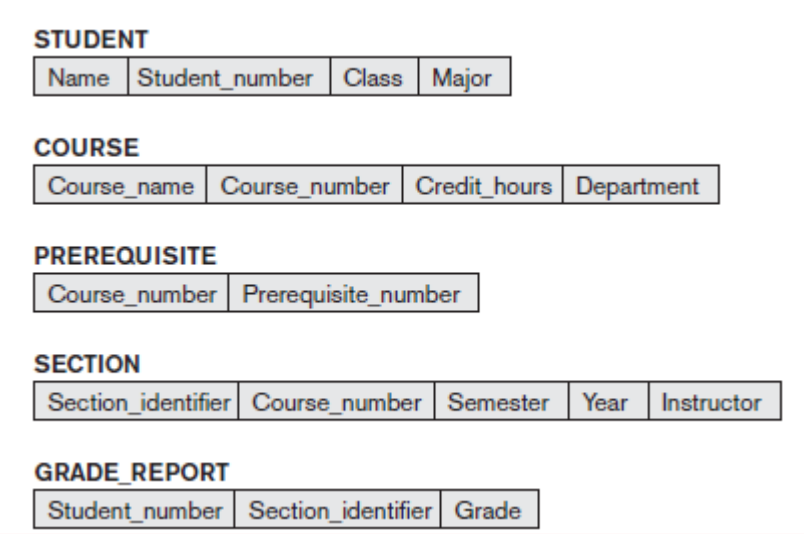
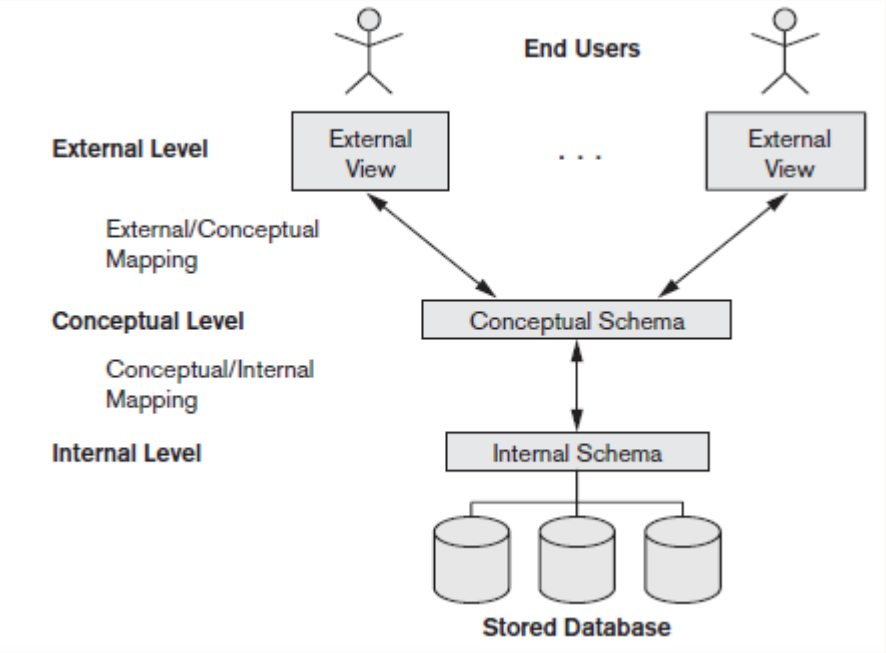


Figure 2.1
Schema diagram for the database

..continued

- The distinction between database schema and database state is very important.
- When we **define** a new database, we specify its database schema only to the DBMS.
- At this point, the corresponding database state is the ***empty state*** with no data.
- We get the ***initial state*** of the database when the database is first **populated** or **loaded** with the initial data.
- From then on, every time an update operation is applied to the database, we get another database state.
- At any point in time, the database has a ***current state***.
- The DBMS is partly responsible for ensuring that every state of the database is a **valid state**—that is, a state that satisfies the structure and constraints specified in the schema.
- Hence, specifying a correct schema to the DBMS is extremely important and the schema must be designed with utmost care.
- The DBMS stores the descriptions of the schema constructs and constraints—also called the **meta-data**—in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to.
- The schema is sometimes called the **intension**, and a database state is called an **extension** of the schema.

Three-Schema Architecture: Separating Levels



The three-schema architecture aims to separate user applications from the physical database by defining schemas at three distinct levels:

1. The **internal 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 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 data model

Data Independence: Changing Schemas Without Impact

Impact

Data independence refers to the ability to change the schema at one level of the database system without needing to change the schema at a higher level. This separation enhances flexibility and maintainability.

The three-schema architecture can be used to further explain the concept of **data independence**

Logical Data Independence

- Change the ***conceptual schema*** without affecting external external schemas or programs.
- We may change the conceptual schema to expand the the database (by adding a record type or data item), to to change constraints, or to reduce the database (by removing a record type or data item).
- In the last case, external schemas that refer only to the the remaining data should not be affected.
- Only the view definition and the mappings need be changed changed in a DBMS that supports logical data independence.

Physical Data Independence

- Change the ***internal schema*** without affecting the conceptual schema.
- Hence, the external schemas need not be changed as well. as well.
- Changes to the internal schema may be needed because because some physical files had to be reorganized-for for example, by creating additional access structures-to structures-to improve the performance of retrieval or or update.
- If the same data as before remains in the database, we we should not have to change the conceptual schema schema

Database Languages

- The DBMS must provide appropriate languages and interfaces for each category of users.
- Once the design of a database is completed and a DBMS is chosen to implement the database, the first step is to specify conceptual and internal schemas for the database and any mappings between the two

DBMS Languages: Defining the Database

Data Definition Language (DDL)

- Used by DBAs and designers to define both conceptual and internal schemas.
- The DBMS uses a DDL compiler to process DDL statements and store the schema description in the catalog.

View Definition Language (VDL)

- 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

Storage Definition Language (SDL)

- Specifies the internal schema.
- In most relational DBMSs, SDL is replaced by functions, parameters, parameters, and specifications related to storage, allowing control control over indexing choices and data mapping.

Data Manipulation Language (DML)

- Typical manipulations include retrieval, insertion, deletion, and and modification of the data.
- The DBMS provides a set of operations or a language called the data called the data manipulation language (DML)

Data Manipulation Language (DML)

- In current DBMSs, the preceding types of languages are usually ***not considered distinct languages***; rather, a comprehensive integrated language is used that includes constructs for conceptual schema definition, view definition, and data manipulation.
- Storage definition is typically kept separate, since it is used for defining physical storage structures to fine-tune the performance of the database system, which is usually done by the DBA staff.

SQL as an Example

- A typical example of a comprehensive database language is the SQL relational database language , which represents a combination of DDL, VDL, and DML, as well as statements for constraint
- specification, schema evolution, and other features.
- The SDL was a component in early versions of SQL but has been removed from the language to keep it at the conceptual and external levels only.

DML Types: High-Level and Low-Level

High-Level DML

- A **high-level** or **nonprocedural** DML can be
- used 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.
- In the latter case, DML statements must be identified within the program so that they can be extracted by a precompiler and processed by the DBMS.
- Highlevel DMLs, such as SQL, can specify and retrieve many records in a single DML
- statement; therefore, they are called **set-at-a-time** or **set-oriented** DMLs.
- A query in a high-level DML often specifies *which* data to retrieve rather than *how* to retrieve it;
- therefore, such languages are also called **declarative**.

Low-Level DML

- A **lowlevel** 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.
- Therefore, it needs to use programming language constructs, such as looping, to retrieve and process each record from a set of records.
- Low-level DMLs are also called **record-at-a-time** DMLs because of this property.
- DL/1, a DML designed for the hierarchical model, is a low-level DML that uses commands such as GET UNIQUE, GET NEXT, or GET NEXT WITHIN PARENT to
- navigate from record to record within a hierarchy of records in the database

- Whenever DML commands, whether high level or low level, are embedded in a general-purpose programming language, that language is called the **host language** and the DML is called the **data sublanguage**.
- On the other hand, a high-levelDML used in a standalone interactive manner is called a **query language**.
- In general, both retrieval and update commands of a high-level DML may be used interactively and are hence considered part of the query language.
- Casual end users typically use a high-level query language to specify their requests, whereas programmers use the DML in its embedded form.
- For naive and parametric users, there usually are **user-friendly interfaces** for interacting with the database; these can also be used by casual users or others who do not want to learn the details of a high-level query language

DBMS Interfaces: Providing User Access

User-friendly interfaces provided by a DBMS may include the following:

1. **Menu-Based Interfaces**
2. **Forms-Based Interfaces**
3. **Graphical User Interfaces (GUIs)**
4. **Natural Language Interfaces**
5. **Interfaces for Parametric Users**
6. **Speech Input and Output**
7. **Interfaces for the DBA.**

Menu-Based Interfaces

- 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; rather, the query is composed step-by step by picking options from a menu that is displayed by the system.
- Pull-down menus are a very popular technique in **Web-based user interfaces**.
- They are also often used in **browsing interfaces**, which allow a user to look through the contents of a database in an exploratory and unstructured manner.

Forms-Based Interfaces

- Display forms for users to input data or retrieve matching data.
- Forms are typically designed for naive users and canned transactions.
- Many DBMSs offer **forms specification languages** which are special languages that help programmers specify such forms. S
- SQL*Forms is a form-based language
- Oracle Forms provides an extensive set of features to design and build applications using forms

Graphical User Interfaces (GUIs) (GUIs)

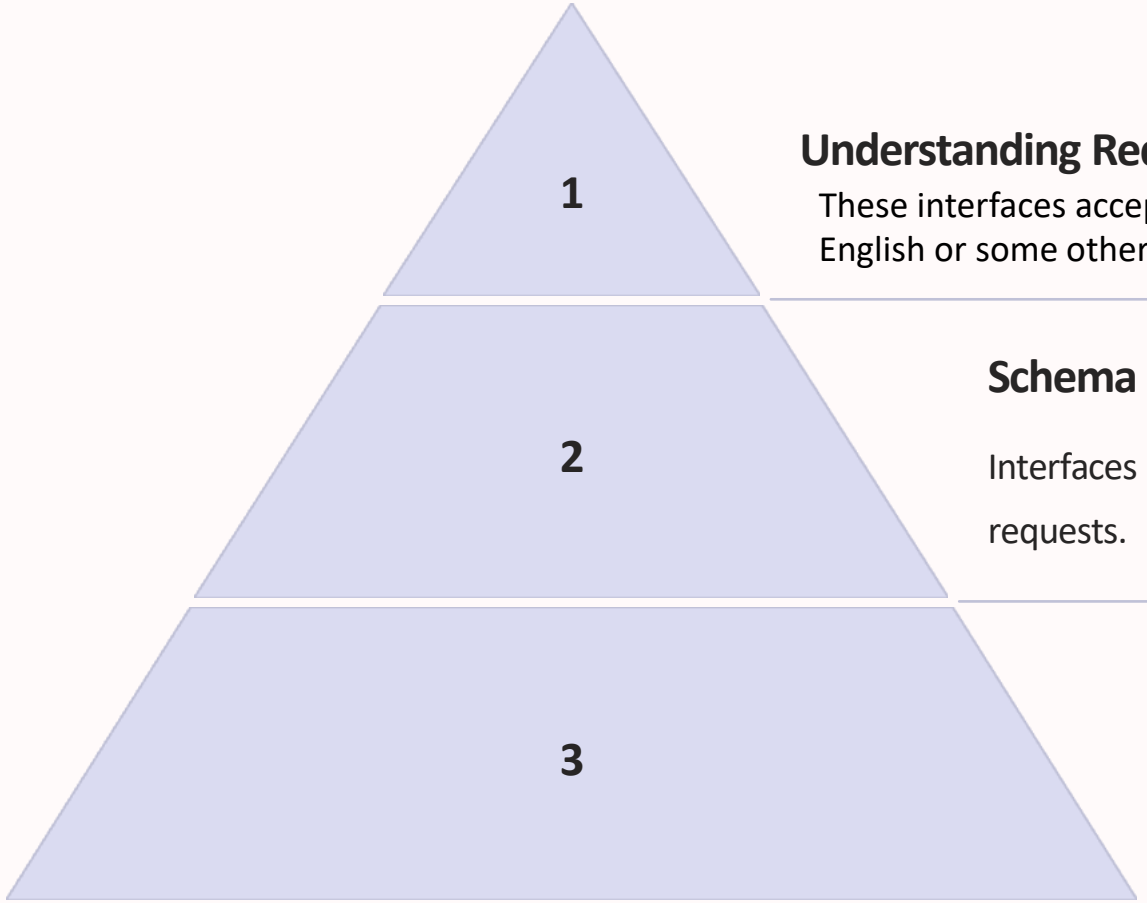
Diagrammatic Schema

- 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

Pointing Devices

Most GUIs utilize pointing devices like mice to select parts parts of the displayed schema schema diagram.

Natural Language Interfaces



Understanding Requests

These interfaces accept requests written in English or some other language and attempt to *understand* them.

Schema and Dictionary

Interfaces use a schema and dictionary to interpret natural language requests.

requests.

Generating Queries

Successful interpretation generates a high-level query for processing.

processing.

Interfaces for Specialized Users

Parametric Users

- Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly.
- For example, a teller is able to use single function keys to invoke routine and repetitive transactions such as account deposits or withdrawals, or balance inquiries
- Systems analysts and programmers design and implement a special interface for each known class of naive users.
- Usually a small set of abbreviated commands is included, with the goal
- of minimizing the number of keystrokes required for each request

DBA Interfaces

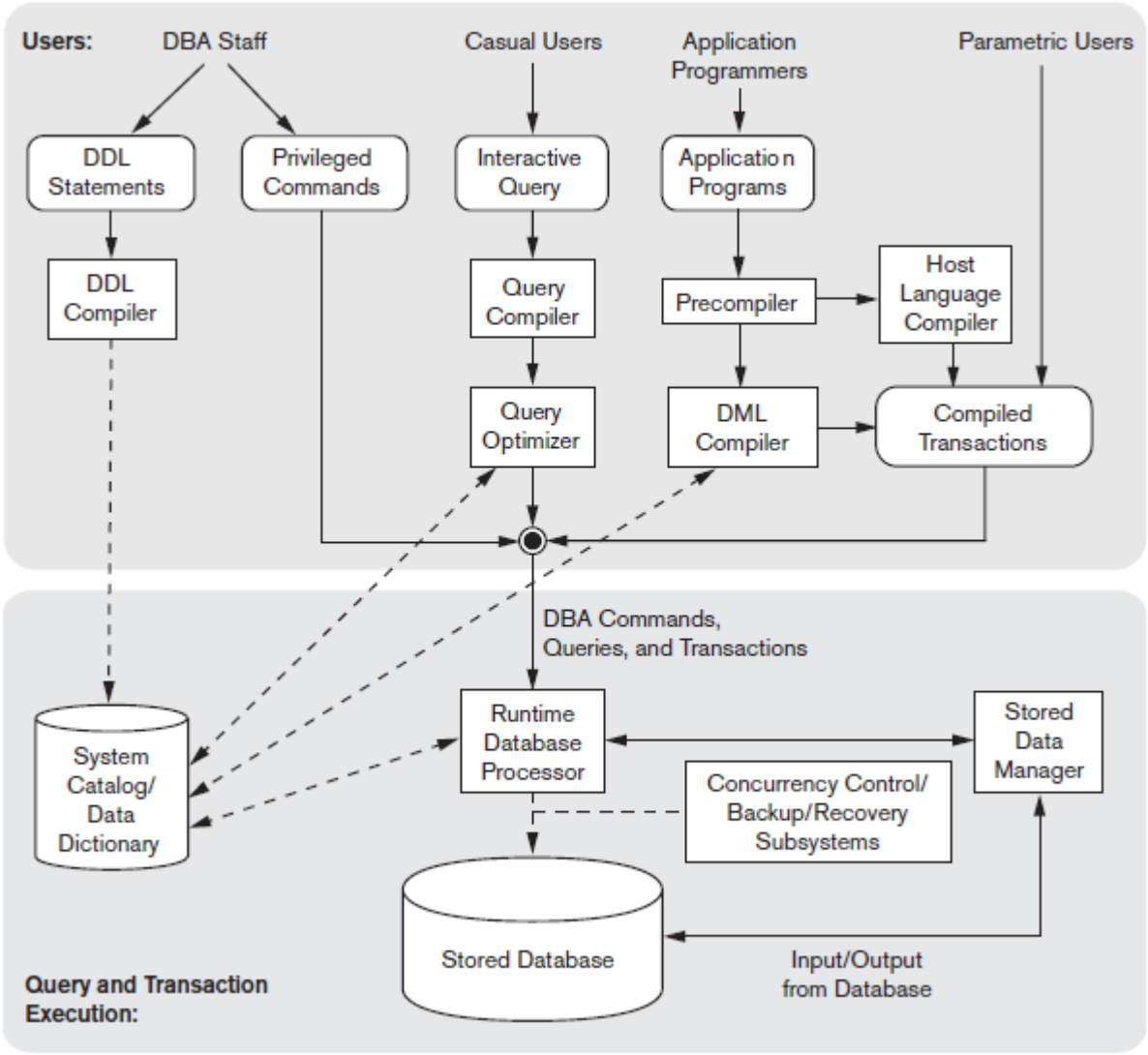
- Most database systems contain privileged commands
- that can be used only by the DBA staff.
- These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

Speech Input and Output

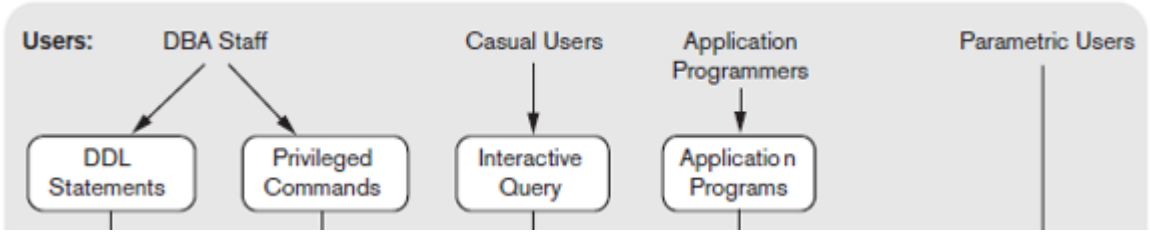
- Limited use of speech as an input query and speech as an answer to a question or result of a request
- The speech input is detected using a library of predefined words and used to set up the parameters that are supplied to the queries.
- For output, a similar conversion from text or numbers into speech takes place.
- Eg - inquiries for telephone directory, flight arrival/departure, and credit card account information

The Database System Environment

- A DBMS is a complex software system. In this section we discuss the types of software components that constitute a DBMS and the types of computer system software with which the DBMS interacts.
- The figure illustrates, in a simplified form, the typical DBMS components.
- The figure is divided into two parts.
- The top part of the figure refers to the various users of the database environment and their interfaces.
- The lower part shows the internals of the DBMS responsible for storage of data and processing of transactions



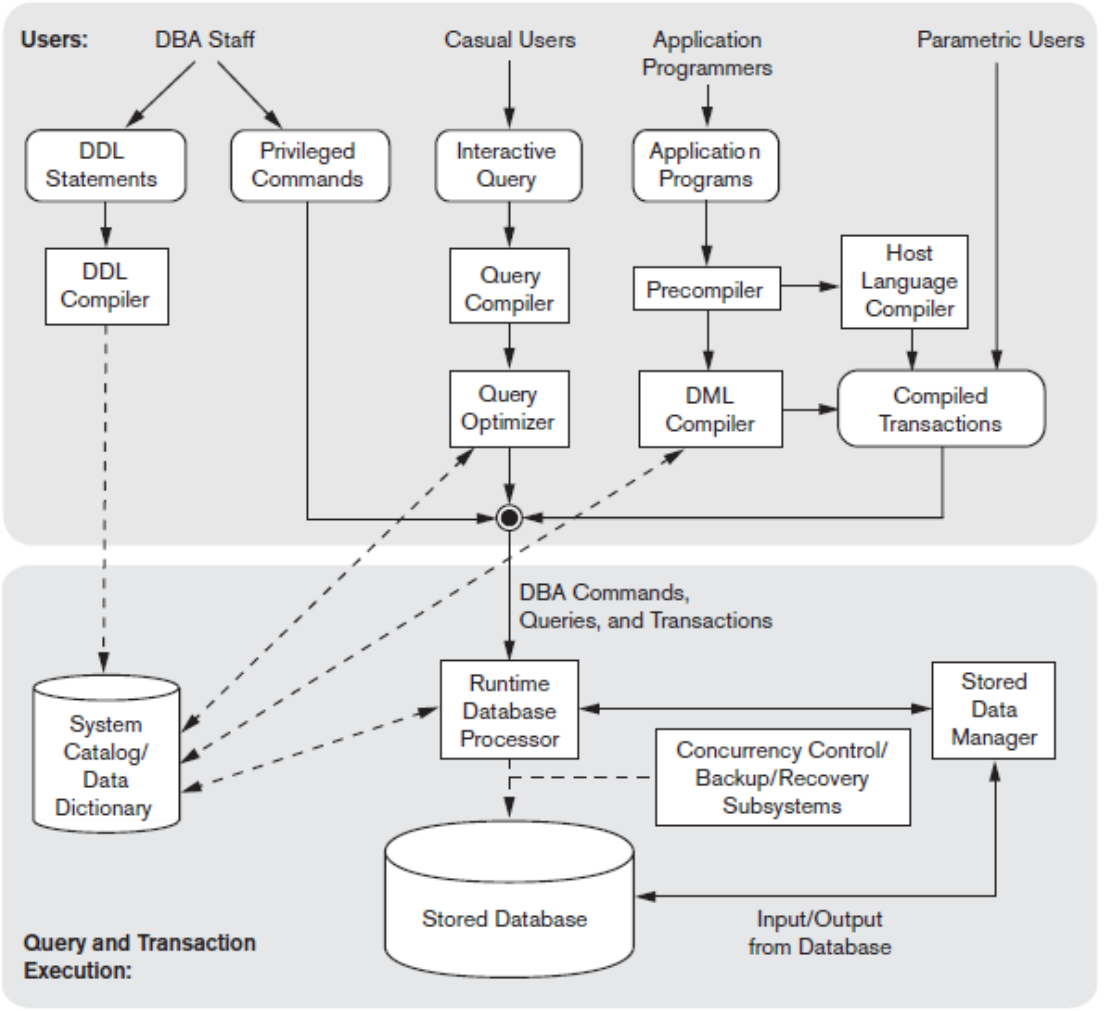
1. Users



- The upper part shows interfaces for DBA staff, casual users who work with interactive interfaces to formulate queries
- Application programmers create programs using some host programming languages
- Parametric users who do data entry work by supplying parameters to predefined transactions.
- The DBA staff works on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands
- Casual users and persons with occasional need for information from the database interact using an interface called the **interactive query** interface

DDL compiler

- 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 and sizes of files, 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, which can then look up the catalog information as needed.



Query compiler

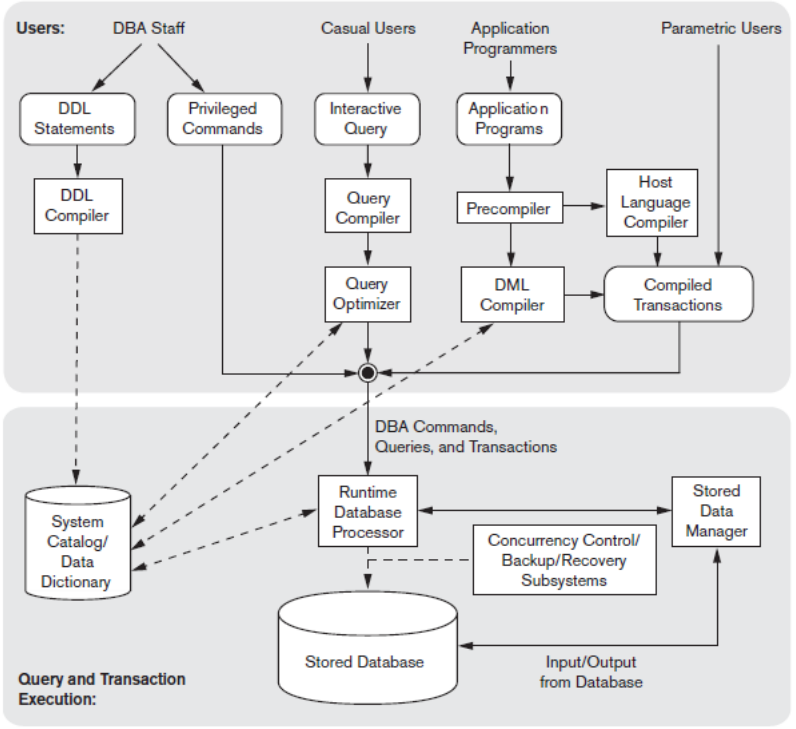
- The **interactive queries** are parsed and validated for correctness of the query syntax, the names of files and data elements, and so on by a **query compiler** that compiles them into an internal form.
- This internal query is subjected to query optimization

Query optimizer

- The **query optimizer** is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution.
- It consults the system catalog for statistical and other physical information about the stored data and generates **executable code** that performs the necessary operations for the query and makes calls on the runtime processor

Precompiler

- The **precompiler** extracts DML commands from an application program written in a host programming language.
- These commands are sent to the DML compiler
- The rest of the program is sent to the host language compiler



DML Compiler

- It collects the DML commands for compilation into object code for database access.

Host language compiler

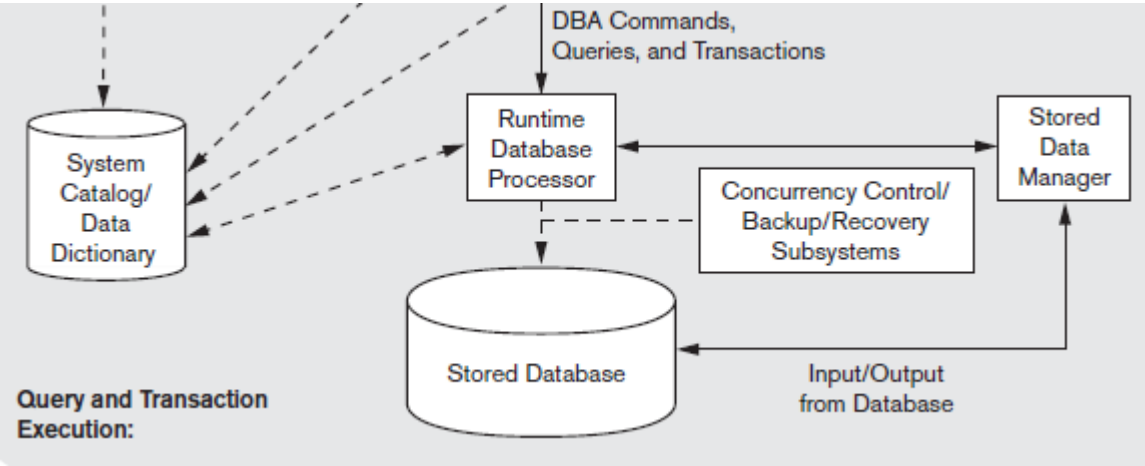
- It collects rest of the programs excluding DML Commands to produce object code

Compiled transaction/canned transaction

• Object codes commands and rest of the program are linked forming a canned transaction whose executable code include call to runtime database processor
• Canned transaction are executed repeatedly by parametric users who simply supply parameter to transaction
• Each execution is a separate transaction

Lower part of architecture includes

1. Runtime database processor
2. Stored data manager
3. Concurrency control/Backup recovery subsystems
4. System catalog/Data dictionary



1. Runtime database processor

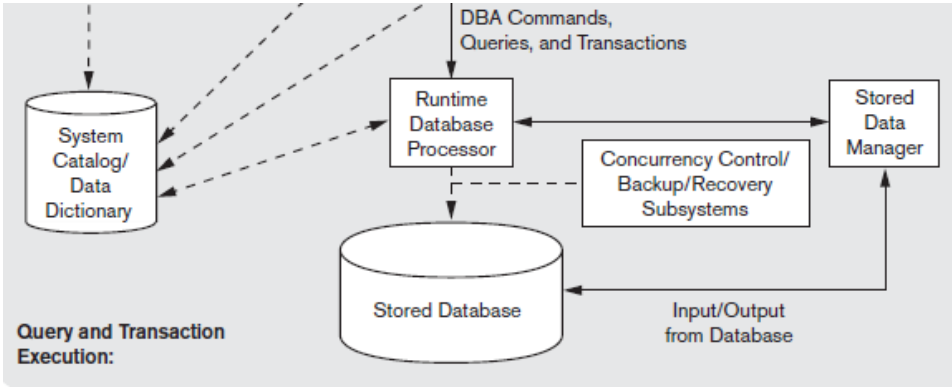
- It executes privileged commands, the executable query plans and canned transactions with runtime parameters
- It handles database access at runtime
- It works with the system catalog and may update it with statistics
- It also works with stored data manager

2. Stored data manager

- It controls the access to DBMS information that is stored in disk
- It uses basic operating system services for carrying out low level input/output operations between disk and main memory

3. Concurrency control/ Backup recovery subsystems

- They are integrated into the working of the runtime database processor for purposes of transaction management
- Backup system creates a backup copy of the database
- Backup copy can be used to restore the database in case of system failure



4.System Catalog/ Data Dictionary

- It store meta-data about structure of database in particular schema of database

Database System Utilities

In addition to possessing the software modules just described, most DBMSs have **database utilities** that help the DBA manage the database system.
Common utilities have the following types of functions:

■ **Loading.** A loading utility is used to load existing data files into the database.
■ **Backup.** A backup utility creates a backup copy of the database.
- 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 to improve performance.

Other utilities may be available for sorting files, handling data compression, monitoring access by users, interfacing with the network, and performing other functions.

Tools, Application Environments, and Communications Facilities

- **CASE** tools are used in the design phase of database systems
- **Data dictionary** (Data repository system) stores other information, such as design decisions,
- usage standards, application program descriptions, and user information.
- Such a system is also called an **information repository**.
- Data dictionary utility is similar to the DBMS catalog, but it includes a wider variety of information and is accessed mainly by users rather than by the DBMS software.

Application development environments provide an environment for developing database applications and include facilities that help in many facets of database systems, including database design, GUI development, querying and updating, and application program development
Eg -PowerBuilder (Sybase) or JBuilder (Borland)

- The DBMS also needs to interface with **communications software**, whose function is to allow users at locations remote from the database system site to access the database through computer terminals, workstations, or personal computers.
- These are connected to the database site through data communications hardware such as Internet routers, phone lines, long-haul networks, local networks, or satellite communication devices.
- Many commercial database systems have communication packages that work with the DBMS.
- The integrated DBMS and data communications system is called a **DB/DC** system