Relational Database Management System 116U01C403

Module 1

Jan 2024-May 2024

Introduction (5)

- **Introduction**
- **Characteristics of databases**
- **Comparison of File system and Database approach**
- **Users of Database system**
- **Concerns when using an enterprise database**
- **Data Independence**
- **DBMS system architecture**
- **Database Administrator**

Introduction

- What is data, database, DBMS:

- **Data:** Known facts that can be recorded and have an implicit meaning, raw
- **Database:** a highly organized, interrelated, and structured set of data about a particular enterprise
  - Controlled by a database management system (DBMS)
- **DBMS**
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- **Database systems:** used to manage collections of data that are
  - Highly valuable
  - Relatively large
  - Accessed by multiple users and applications, often at the same time
- **A modern database system:** a complex software system whose task is to manage a large, complex collection of data.

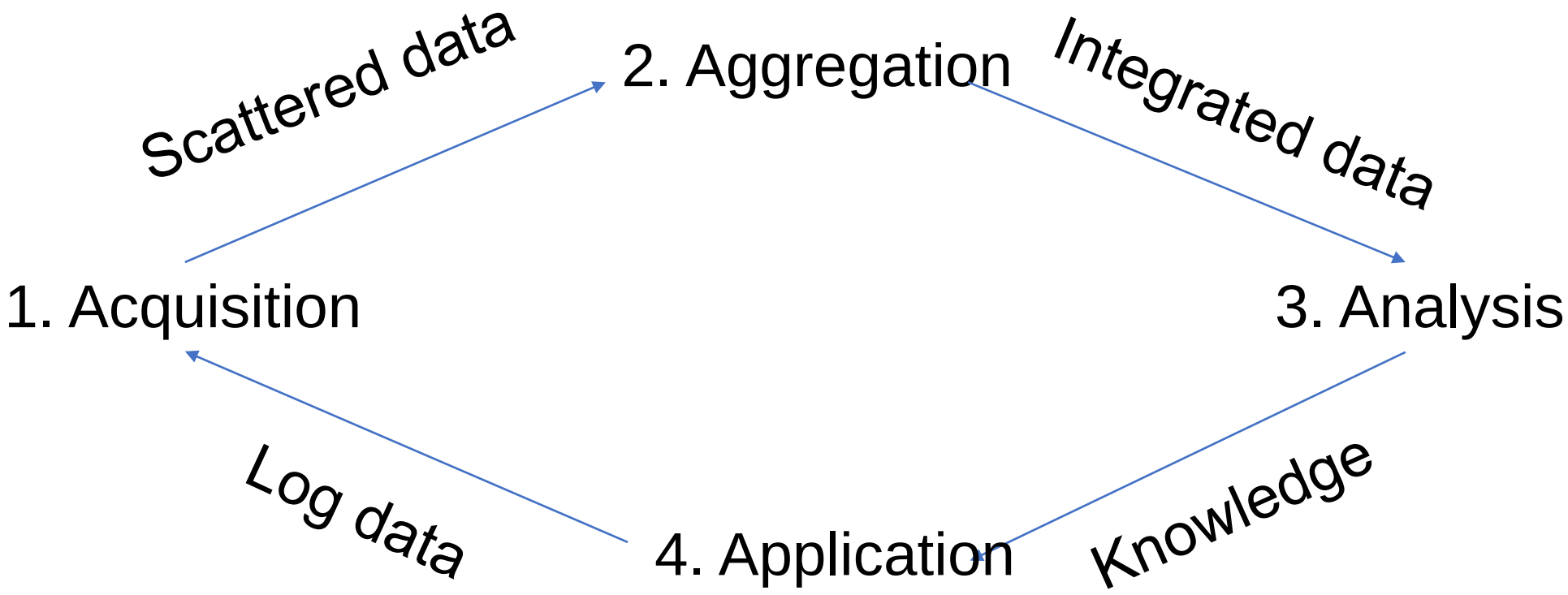
Types of Databases and Database Applications

- Traditional applications:
  - Numeric and textual databases
- More recent applications:
  - Multimedia databases
  - Geographic Information Systems (GIS)
  - Biological and genome databases
  - Data warehouses
  - Mobile databases
  - Real-time and active databases

Importance of “big data”

- Government
- Private Sector
  - Walmart handles more than 1 million customer transactions every hour, which is imported into databases estimated to contain more than 2.5 petabytes of data
  - Facebook handles 40 billion photos from its user base
  - Falcon Credit Card Fraud Detection System protects 2.1 billion active accounts world-wide
- Science
  - Large Synoptic Survey Telescope will generate 140 Terabyte of data every 5 days
  - Biomedical computation like decoding human Genome and personalized medicine

Lifecycle of Data: 4 “A”s



Basic Definitions

Full Page

- **Database:**
  - A collection of related data.
- **Data:**
  - Known facts that can be recorded and have an implicit meaning.
- **Mini-world:**
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Database Management System (DBMS):**
  - A software system to facilitate the creation and maintenance of a computerized database.
- **Database system:**
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

File system

- **Large amount of data needed for storing and processing**
- **Volatile nature of data :** while the program is running the out put will be stored/ displayed
- **Sharing of information between various programs**
- **Records/ structure/ union/ vectors**
- **Sequential access/ Random access**

Facilities provided by DBMS

- *Define* a particular database in terms of its data types, structures, and constraints
- *Construct* or load the initial database contents on a secondary storage medium
- *Manipulating* the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- *Processing* and *sharing* by a set of concurrent users and application programs – yet, keeping all data valid and consistent
- Protection or security measures to prevent unauthorized access
- "Active" processing to take internal actions on data
- *Presentation* and *visualization* of data
- *Maintenance* of the database and associated programs over the lifetime of the database application

Applications interact with a database by generating

- **Queries:** that access different parts of data and formulate the result of a request

- **Transactions**: that may read some data and “update” certain values or generate new data and store that in the database

Characteristics of the Database Approach

- **Self-describing nature of a database system:**
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data***.
  - This allows the DBMS software to work with different database applications.
- **Insulation between programs and data:**
  - Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs
    - E.g., ADTs

Characteristics of the Database Approach (..contd)

- **Data abstraction:**
  - A **data model** is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- **Support of multiple views of the data:**
  - Each user may see a different view of the database, which describes **only** the data of interest to that user.

Characteristics of the Database Approach (..contd)

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each transaction is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications; allows hundreds of concurrent transactions to execute per second.

Database users

- **Users may be divided into**
  1. Those who actually use and control the database content, and those who design, develop and maintain database applications (called "*Actors on the Scene*")

  2. Those who design and develop the DBMS software and related tools, and the computer systems operators (called "*Workers Behind the Scene*").

Database users: Actors on the scene

- **Database administrators**
  - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.
- **Database designers**
  - Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

Database users: Actors on the scene

- **End-users:** They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
  - **Casual**: access database occasionally when needed
  - **Naïve** or parametric: they make up a large section of the end-user population.
    - They use previously well-defined functions in the form of "canned transactions" against the database.
    - Users of mobile apps mostly fall in this category
    - Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
    - Social media users post and read information from websites

Database users: Actors on the scene

- **Sophisticated:**
  - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
  - Many use tools in the form of software packages that work closely with the stored database.
- **Stand-alone:**
  - Mostly maintain personal databases using ready-to-use packaged applications.
  - An example is the user of a tax program that creates its own internal database.
  - Another example is a user that maintains a database of personal photos and videos.

Database users: Actors on the scene

- **System analysts and application developers**
  - System analysts: They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.
  - Application programmers: Implement the specifications developed by analysts and test and debug them before deployment.
  - Business analysts: There is an increasing need for such people who can analyze vast amounts of business data and real-time data (“Big Data”) for better decision making related to planning, advertising, marketing etc.

Database users: Actors behind the scene

- **System designers and implementors:**
  - Design and implement DBMS packages in the form of modules and interfaces and test and debug them. The DBMS must interface with applications, language compilers, operating system components, etc.
- **Tool developers**:
  - Design and implement software systems called tools for modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc. that facilitate building of applications and allow using database effectively.
- **Operators and maintenance personnel:**
  - They manage the actual running and maintenance of the database system hardware and software environment.

Database users

- Application Programmer: interact with system through DML calls
- Sophisticated users: form requests in a database query language
- Specialized users: write specialized database applications that do not fit into the traditional database processing framework
- Naïve users: invoke one of the permanent application programs that have been written permanently

Database Administrator

- Coordinates all activities related to database systems
- Has good understanding of the enterprise’s information
- Administrator's duties include:
  - Schema definition
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting user authorities to access database
  - Specifying integrity constraints
  - Acting as liaison with users
  - Monitoring performance and responding to changes in requirements

Advantages of Using the Database Approach

- Controlling redundancy in data storage and in development and maintenance efforts.
  - Sharing of data among multiple users.
- Restricting unauthorized access to data. Only the DBA staff uses privileged commands and facilities.
- Providing persistent storage for program Objects
  - E.g., Object-oriented DBMSs make program objects persistent
  - Providing storage structures (e.g. indexes) for efficient query
- Providing optimization of queries for efficient processing
- Providing backup and recovery services
- Providing multiple interfaces to different classes of users
- Representing complex relationships among data
- Enforcing integrity constraints on the database
- Drawing inferences and actions from the stored data using deductive and active rules and triggers

Advantages of Using the Database Approach

- Potential for enforcing standards:
  - **Standards** refer to data item names, display formats, screens, report structures, meta-data (description of data), Web page layouts, etc.
- Reduced application development time:
  - Incremental time to add each new application is reduced.
- Flexibility to change data structures:
  - Database structure may evolve as new requirements are defined.
- Availability of current information:
  - Extremely important for on-line transaction systems such as shopping, airline, hotel, car reservations.
- Economies of scale:
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

Limitations of the Database Approach

- Main inhibitors (costs) of using a DBMS:
  - High initial investment and possible need for additional hardware
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions
- When a DBMS may be unnecessary:
  - If the database and applications are simple, well defined, and not expected to change
  - If access to data by multiple users is not required
- When a DBMS may be infeasible
  - In embedded systems where a general-purpose DBMS may not fit in available storage
- When no DBMS may suffice:
  - If there are stringent real-time requirements that may not be met because of DBMS overhead (e.g., telephone switching systems)
  - If the database system is not able to handle the complexity of data because of modeling limitations
  - If the database users need special operations not supported by the DBMS

Data Model

- **Data Model:**
  - A set of concepts to describe the ***structure*** of a database, the ***operations*** for manipulating these structures, and certain ***constraints*** that the database should obey.
- **Data Model Structure and Constraints:**
  - Constructs are used to define the database structure
  - Constructs typically include ***elements*** (and their ***data types***) as well as groups of elements (e.g. ***entity, record, table***), and ***relationships*** among such groups
  - Constraints specify some restrictions on valid data; these constraints must be enforced at all times
- **Data Model Operations:**
  - These operations are used for specifying database retrievals and updates by referring to the constructs of the data model.

Operations on the data model may include **basic model** operations (e.g. generic insert, delete, update) and **user-defined** operations (e.g. compute_student_gpa, update_inventory)

Data Model Categories

Conceptual (high-level, semantic) data models:

Provide concepts that are close to the way many users perceive data.

(Also called ***entity-based*** or ***object-based*** data models.)

Physical (low-level, internal) data models:

Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals

Implementation (representational) data models:

Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Data Schemas and Database Instance

Database Schema:

The ***description*** of a database.

Includes descriptions of the database structure, data types, and the constraints on the database.

Schema Diagram:

An ***illustrative*** display of (most aspects of) a database schema.

Schema Construct:

A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.

Data Schemas and Database Instance

Database State / database instance (or occurrence or snapshot):

The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database.

The term *instance* is also applied to individual database components, e.g. *record instance, table instance, entity instance*

Data Schemas and Database Instance

Database State / database instance (or occurrence or snapshot):

Database State:

Refers to the content of a database at a moment in time.

Initial Database State:

Refers to the database state when it is initially loaded into the system.

Valid State:

A state that satisfies the structure and constraints of the database.

Distinction

The ***database schema*** changes very infrequently.

The ***database state*** changes every time the database is updated.

Schema is also called **intension**.

State is also called **extension**.

Data Schemas

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

Database Instance

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	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

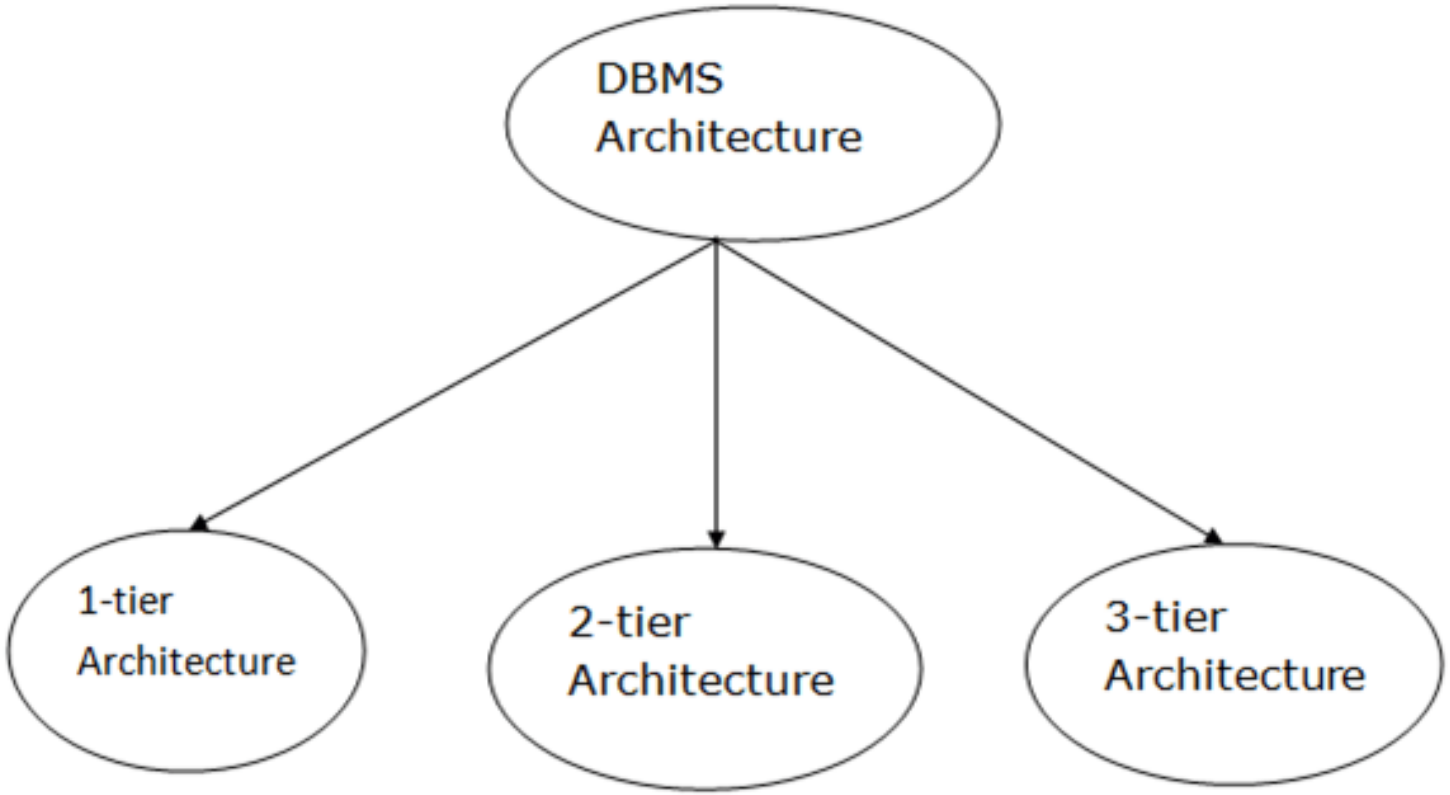
GRADE_REPORT

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

PREREQUISITE

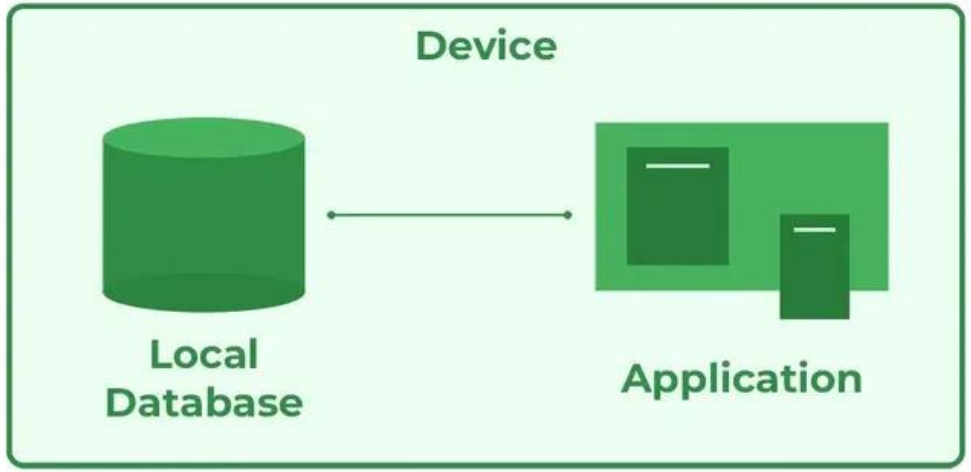
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Types of DBMS Architecture



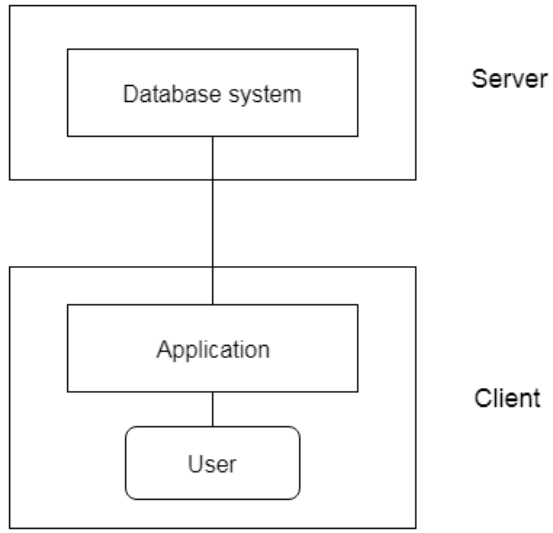
1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.



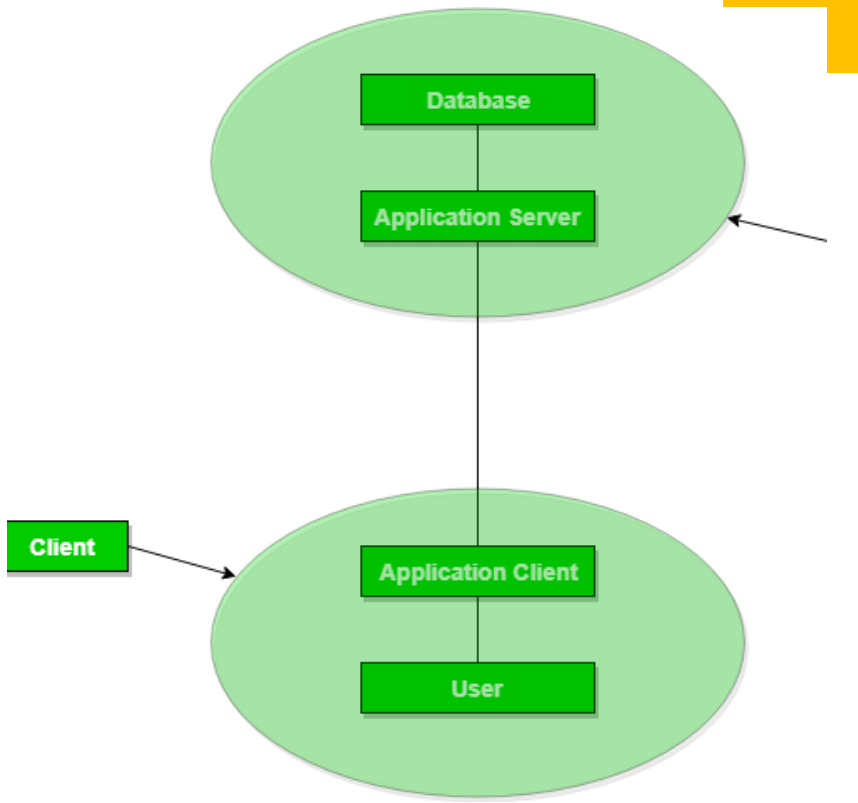
2-Tier Architecture

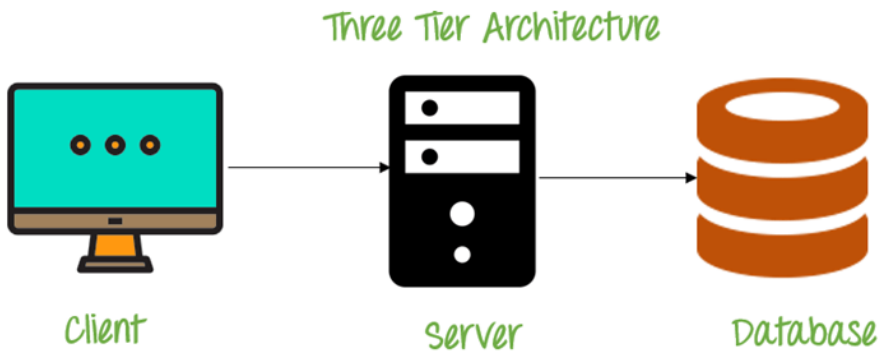
- The 2-tier architecture is similar to a basic client-server model.
- In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.



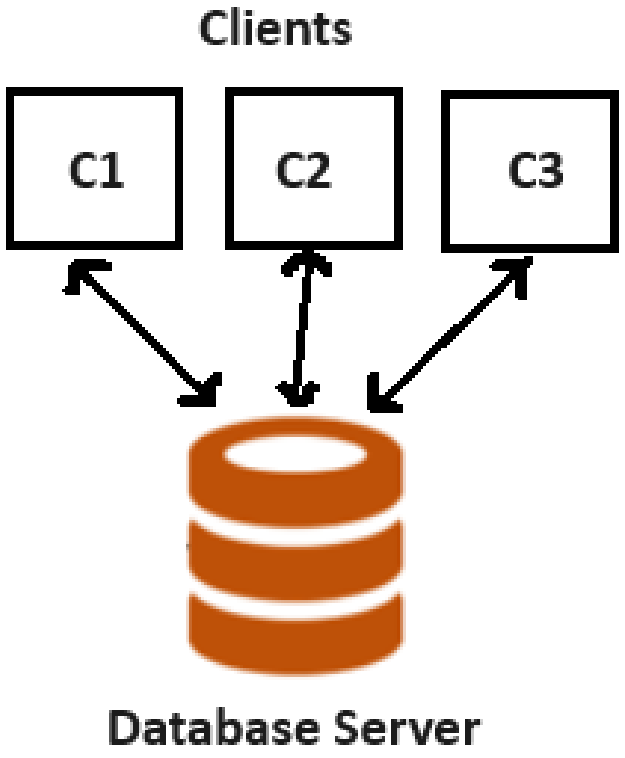
3-Tier Architecture

- The 3-Tier architecture contains another layer between the client and server.
- In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

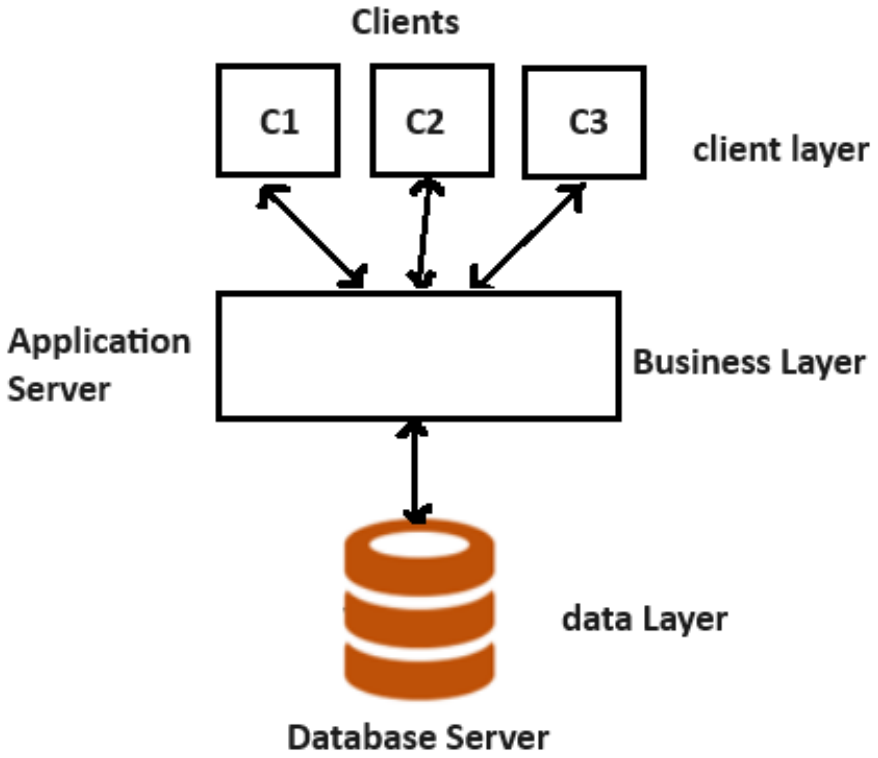




2-Tier


Clients
C1
C2
C3
Database Server

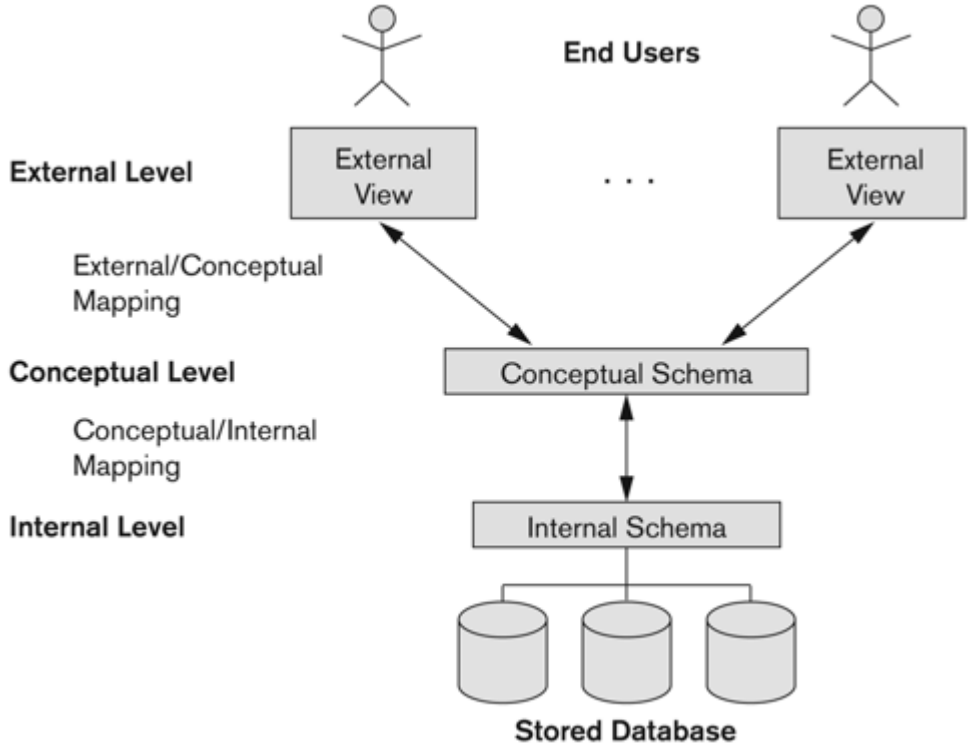
3-Tier


Clients
C1
C2
C3
client layer
Application
Server
Business Layer
data Layer
Database Server

Three schema Architecture

Defines DBMS schemas at ***three*** levels:

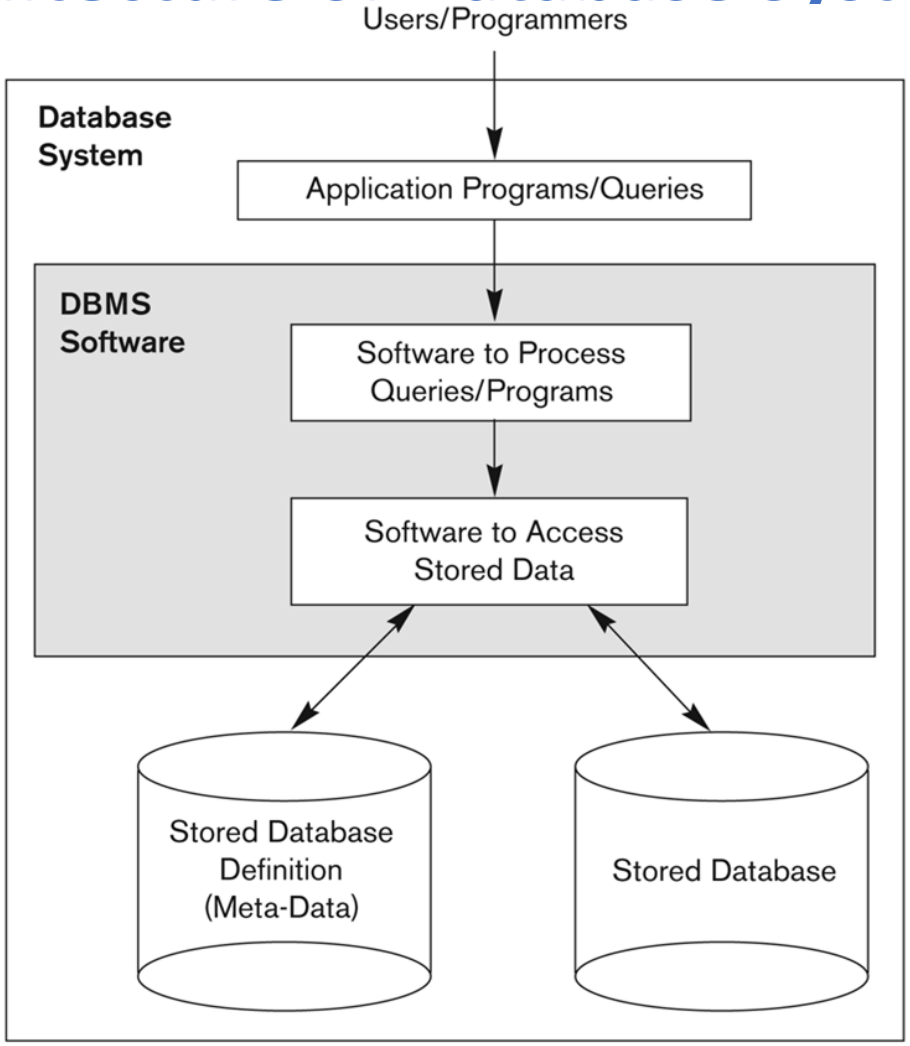
- **Internal schema** at the internal level to describe physical storage structures and access paths (e.g. indexes).
  - Typically uses a **physical** data model.
- **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.
  - Uses a **conceptual** or an **implementation** data model.
- **External schemas** at the external level to describe the various user views.
  - Usually uses the same data model as the conceptual schema.



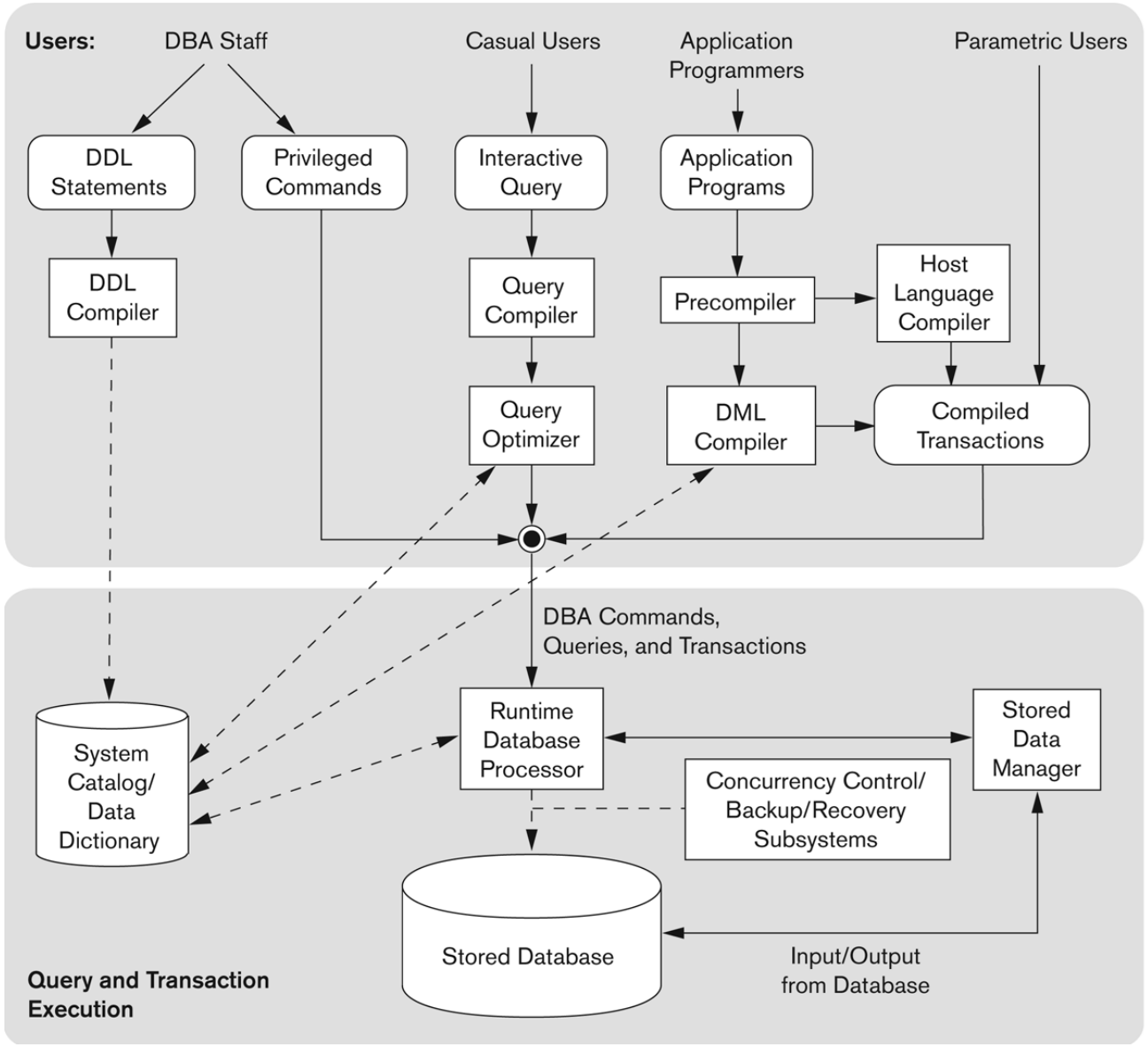
Three schema Architecture

- Mapping is used to transform the request and response between various database levels of architecture.
- Mapping is not good for small DBMS because it takes more time.
- 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.

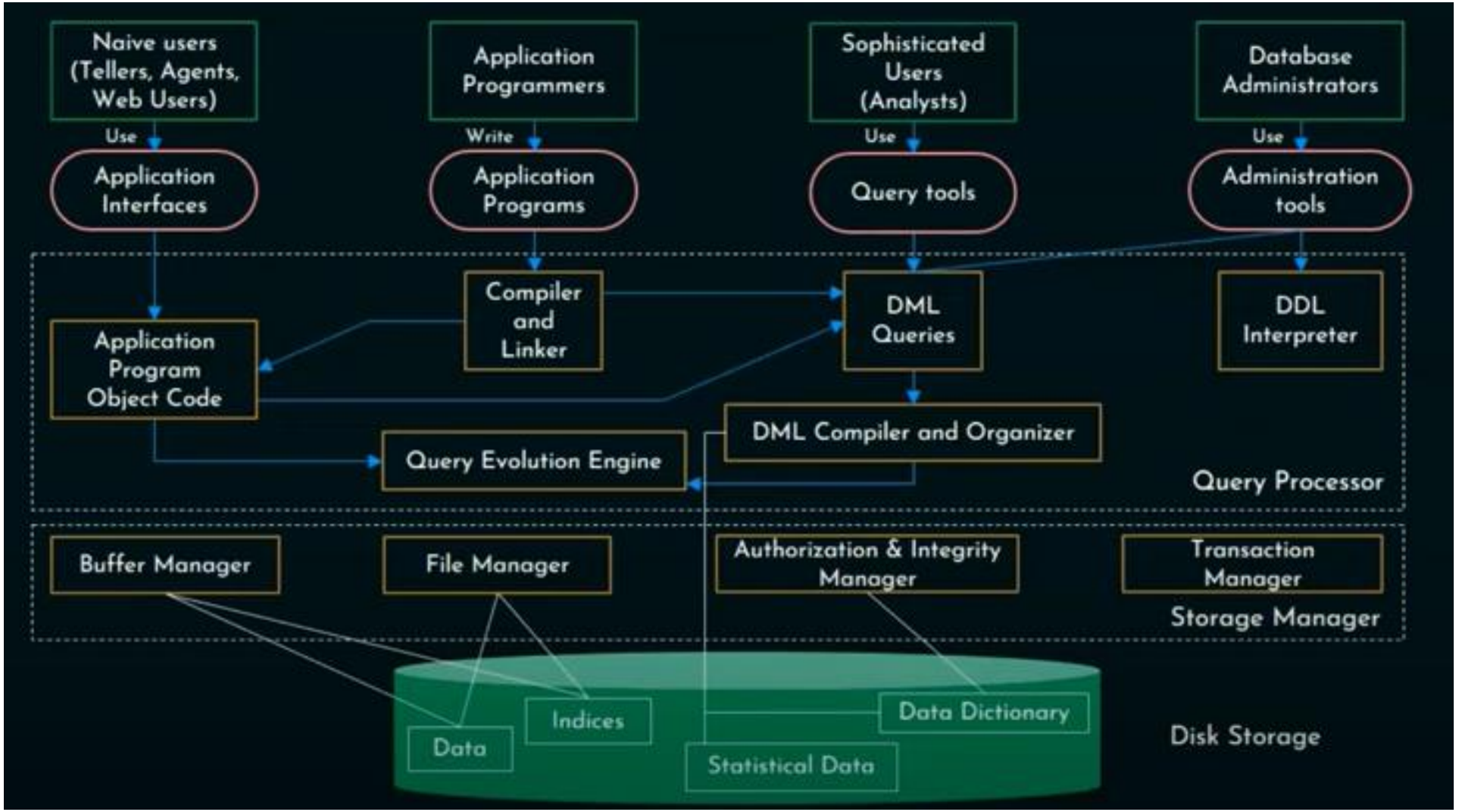
A simplified architecture of Database System



A simplified architecture of Database System

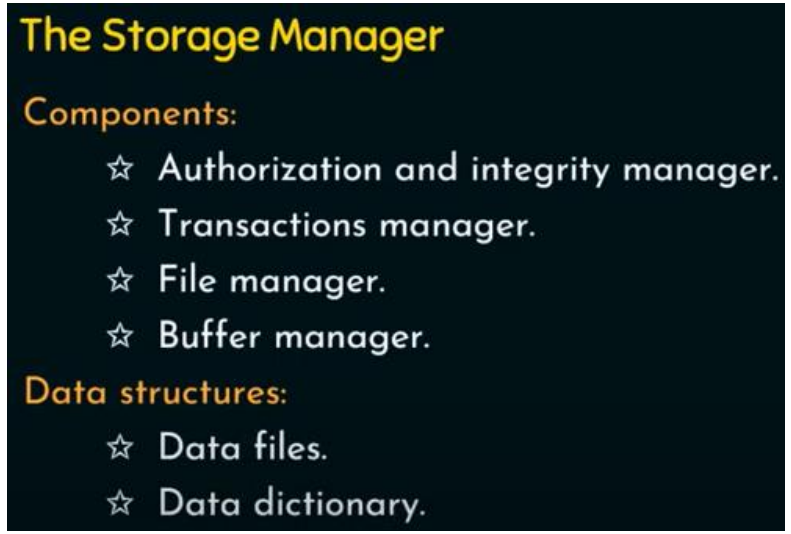


A simplified architecture of Database System



Buffer manager-cache the data on priority. Handle disk storage management

File manager- memory management and data structure allocation



The Query Processor

- ☆ DDL Interpreter.
- ☆ DML Compiler.
- ☆ Query Evaluation Engine.