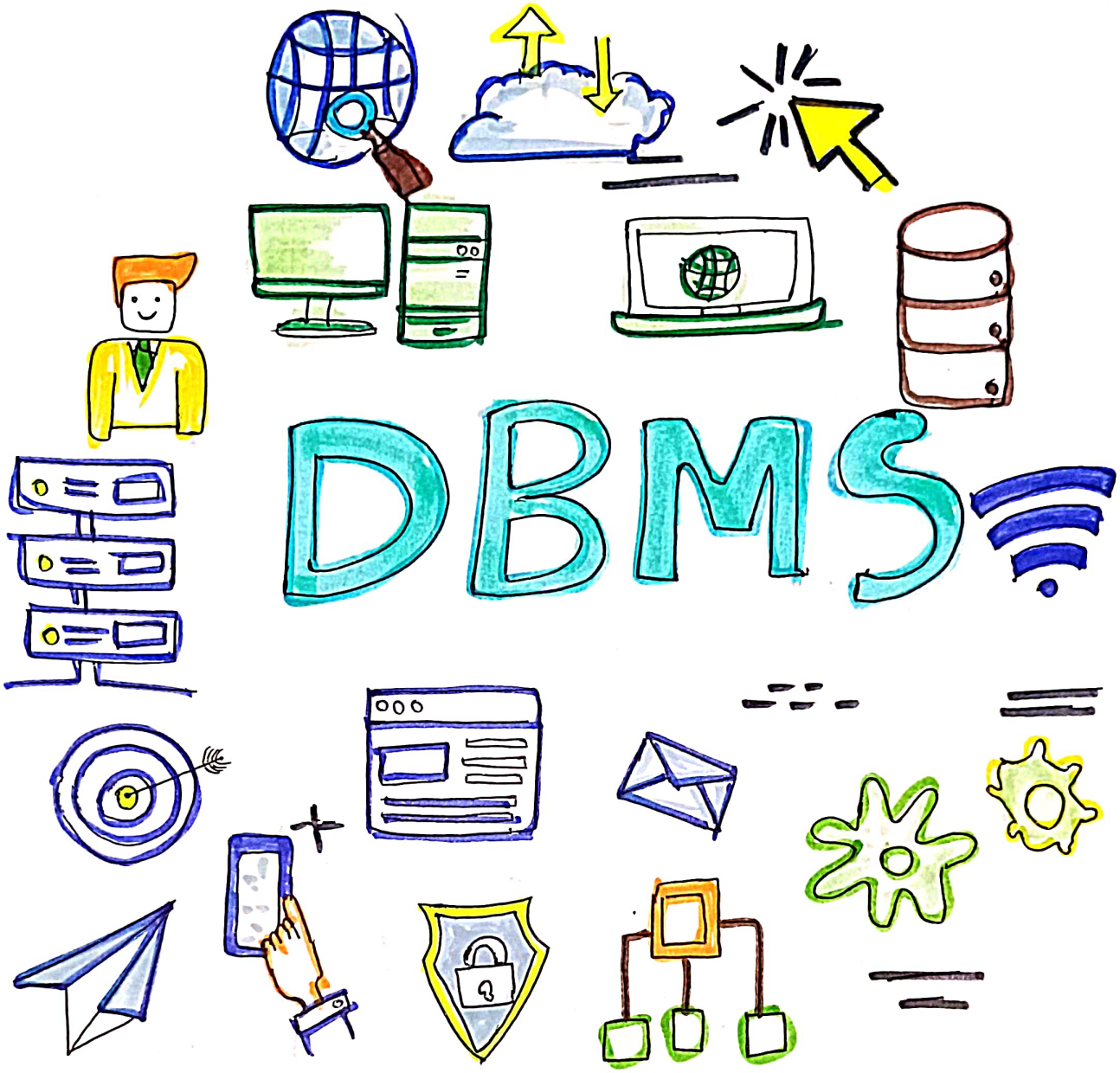


: Himanshu Kr. Singh

DBMS (Important topics : Interview)

- Why DBMS ?
- Key → Primary Key
  → Candidate Key
  → foreign Key
- ACID properties
- ★ Different types of relationship
  - 1:1
  - 1:many
  - many : many
- Normalization & Denormalization
  - ★ Its forms : 1NF, 2NF, 3NF, etc.
- Lock in Database
- Basic SQL commands
- Writing Basic query
  - Insertion / deletion in a table, etc.
- Join & its types
- Indexing

Books :-

Henry F. Korth (Transaction, ER)
Navathe (FD & N.F)
Raghu Ramakrishnan (Query, ER)
Jeffrey D. Ullman (Indexing)

• Database Management System

A database management system (DBMS) is a collection of interrelated data and a set of programs to access those data.

• Primary goal :- Store and retrive ★ convenient and efficient.

• Data (Fact)
↓
Information (Pre processed data)
↓
Database (collection of logically related data or collection of similar records)
↓
DBMS (Set of program [s/w] used to access & update the data in efficient manner.)

• Data Vs Information

- Data usually refer to ra... facts/data.
- Information is organized ... it has some mean... Or
★ Information is processed ... and actions are ba... cisions

Eg Data - 25, Ram, Jaipur
Information - The age of Ram is 25 who is residing in Nagpur.

• Database - collection of interrelated data.

- A database is shared collection of logically related data designed to meet the information needs of an organization.
- The related information when placed is an organized form makes a database.

Eg University Database.

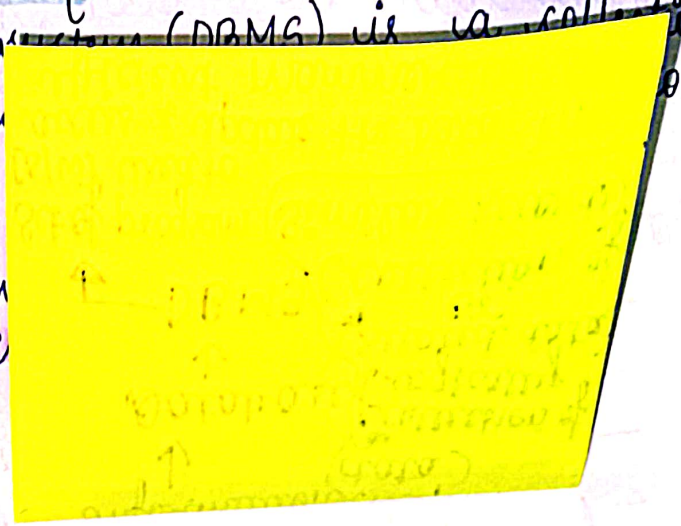
• DBMS

- A database management system is the software system that allows user to define, create and maintain a database and provide controlled access to the data.

• Database Management System

A database management system (DBMS) is a collection of interrelated data and a set of programs to access those data.

• Primary goal :- Store and retrieve database information that is both convenient and efficient.



• Data Vs Information

- Data usually refer to raw data or unprocessed facts/data.
- Information is organized classified data so that it has some meaningful values to the receiver.

Or

* Information is processed data [on which decisions and actions are based].

Eg Data - 25, Ram, Jaipur
Information - The age of Ram is 25 who is residing in Nagpur.

• Database - collection of interrelated data.

- A database is shared collection of logically related data designed to meet the information needs of an organization.
- The related information when placed is an organized form makes a database.

Eg University Database.

• DBMS

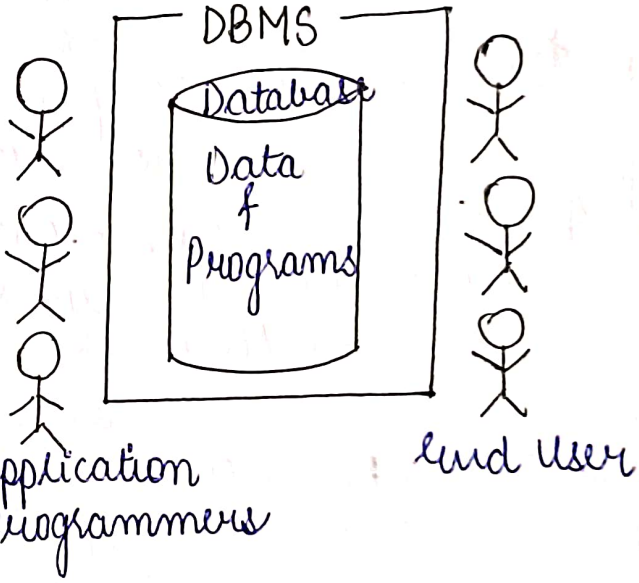
- A database management system is the software system that allows user to define, create and maintain a database and provide controlled access to the data.

- A DBMS is basically a collection of programs that enables users to store, modify, and extract information from a database as per the requirements.

- Operation on Database
  - Add new information.
  - View or retrive stored information
  - Modify or edit the existing
  - Remove or delete the unwanted information
  - Arranging the information. (etc)

Components of Database

- Five major components :-
  - Hardware
  - Software
  - Data
  - Users
  - Procedures
  - Database Access Language (Minor)



- DBMS general purpose software system enabling :-
  - Creation of large disk-resident database.
  - Posting of data retrieval queries in a standard manner.
  - Retrieval of query results efficiently.
  - Concurrent use of system by a large number of users in a consistent manner.
  - Guaranteed availability of data irrespective of system failure.

• File System Vs DBMS

• separation of data & metadata
• flexibility of changing metadata
• program-data independence.

1) Data Redundancy & Inconsistency

File system → wastage of space.

Redundancy: Multiple files created by different programmer often lead to duplication of data.

Eg: If same employee data is stored across multiple files, changes in one file won't reflect in others.

Inconsistency: Data changes in one file may not sync across all file, leading to mismatched information.

DBMS →

- Centralized storage ensure updates are reflected across all associated data.
- constrains like primary keys prevent duplication, maintaining data accuracy.

- DBMS (System development)
  - ↳ less effort required.
  - ↳ concentration on logical level design is enough
  - ↳ components to organize data storage: process queries, manage concurrent access, recovery from failure, manage access control are all available.

2) Difficulty in Accessing data

File system →

- Require manual filtering and conditional formatting for retrieval.
- complex queries (eg finding employees with a salary >50,000) demand manual effort.

DB

- Si... ...ith SQL queries (eg SELECT * FROM Employees WHERE Salary > 50000)
- Query-based access save times and effort even for complex condition.
- ad-hoc query formulation - easy

3) Data Isolation

File System

- Data scattered across various files in different location complicate access.

DBMS

- Centralized database ensures all data is stored in one place, making it easier to access & manage.
- update in one table reflect automatically across dependent tables.

• File System Vs DBMS

1) Data Redundancy & Inconsistency

File system → wastage of space.

Redundancy: Multiple files created by different programmer often lead to duplication of data.

Eg: If same employee data is stored across multiple files, changes in one file won't reflect in others.

Inconsistency: Data changes in one file may not sync across all file, leading to mismatched information.

DBMS →
- separation of data & metadata.
- flexibility of changing metadata.
- program-data independence.

- Centralized storage ensure updates are reflected across all associated data.
- constrains like primary keys prevent duplication, maintaining data accuracy.

• DBMS (System development),

2) Difficulty in Accessing data

File system →
- Require manual filtering and conditional formatting for retrieval.
- complex queries (eg finding employees with a salary >50,000) demand manual effort.

DBMS:-
- Simplifies data retrieval with SQL queries (eg SELECT * FROM Employees WHERE Salary > 50000)
- Query-based access save times and effort even for complex condition.
- ad-hoc query formulation - easy

3) Data Isolation

File System
- Data scattered across various files in different location complicate access.

DBMS
- Centralized database ensures all data is stored in one place, making it easier to access & manage.
- Update in one table reflect automatically across dependent tables.

4) Integrity Problem

File system
- Cannot enforce rule like: Salary should not be 0 employee Id should be unique.
- eg - File system may accept invalid data (eg Negative salary)

DB:-
- Allows integrity constraints such as NOT NULL, UNIQUE, and CHECK to validate data.
- eg CHECK (salary > 0) ensure salary values remain positive.

---

5) Atomicity problem

File system
Partial execution of task leads to inconsistencies

eg A failed transcation (debiting an account without crediting the distination) leaves the system in an invalid state.

DB:-
- Transaction follow ACID properties (Atomicity, consistency, Isolation, Durability).
- If a failure occurs, changes roll back, ensuring "all-or-none" execution.

---

6) Concurrent access anomalies.

File system
- Concurrent access can lead to conflicts (eg two users updating the same record simultaneously).
- Data corruption or loss may occur.

DB:-
- Concurrency control mechanism (eg lock, timestamps) ensure consistency even with simultaneously access.
- A bank system allows multiple users to access accounts without overwriting data.

---

7) Security Problem

File system
- Limited access or no access control.
- Anyone with file access can modify or delete data.

DB
- Implement role based access control (RBAC)
- Data encryption enhance security further.

eg Admin can update sensitive data, while others users have read-only access.

Application of DBMS

1) Banking: all transactions
2) Airlines: reservations, schedules
3) Universities: registration, grades
4) Sales: customers, products, purchases
5) Online retailers: online tracking, customized recommendations.
6) Manufacturing: production, inventory, orders, supply chain
7) Human resources: employee records, salaries, tax, deductions.

Data Model :-

→ Collection of conceptual tools to describe the database at a certain level of abstraction.

A set of concepts to describe the structure of a database, and certain constraints that the database should obey.

Schema :-

The overall description of the database is called the Database schema.

- A schema is defined as an outline or plan that describes the records and relationships existing at the particular level.

Instance :-

The collection of the information stored in the database at a particular moment of time is called an instance of the database.

Note :-

eg (relation)

Database schema = Variable declaration
Database instant = Value of the variable

- Three-Tier Architecture

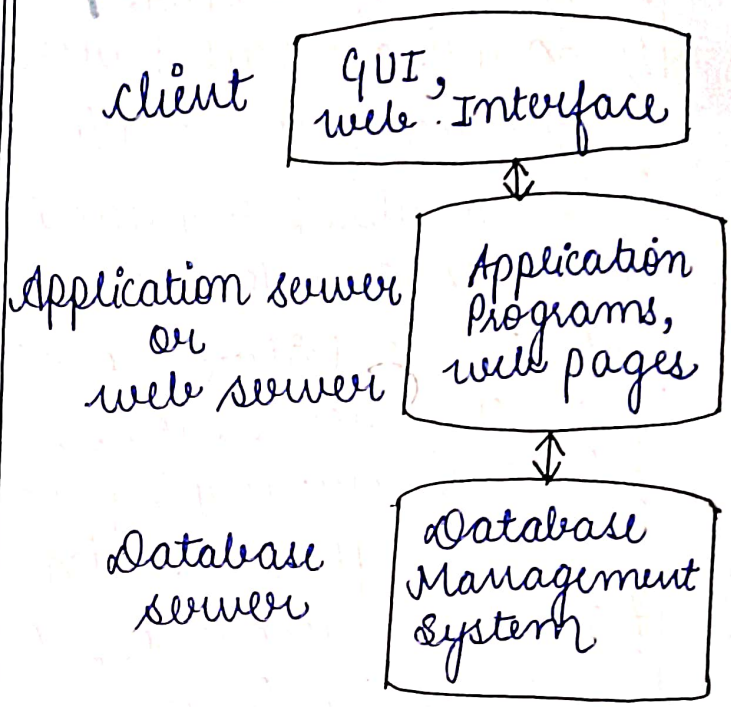
A predominant software architecture for client-server applications with three logical and physical tier.

Tiers :-

a) Client Tier (Presentation Tier)

b) Application Tier (Middleware/ Business logic Tier)

c) Database Tier (Data storage Tier)



Why Three Tier?

- Traditional two-tier architecture (client-server) isn't sufficient for modern needs like web pages and remote access.
- Three tiers improve scalability, reliability, and security.
- Each tier has:
  - Independent infrastructure: can be developed, scaled and updated individually.
  - Faster development and maintenance.

1) Client Tier

→ Role :- Topmost tier; interacts directly with the user.

→ Components :-

- User Interface (UI): collect and display data.
- Frontend technologies :- HTML, CSS, JS, etc.

→ Example -

- Banking applications accessed via a browser or mobile app.

Display account balance and collects inputs like transaction details.

→ Responsibilities :-
- Data collection - Inputs from users (eg fund transfer details)
- Data display - Output from the database (eg. balance)

→ Interaction :-
connects with the application tier.

2) Application Tier

→ Role :- Middle tier ; processes business logic and rules.

→ components :-
- Technologies like Python, Java, PHP, Ruby, etc.
- Use API's to interact with database tier.

→ Responsibilities :-
- Implements business logic (eg fund transfer rules).
- Processes data request from the client tier.
- Communicate with the database tier via API calls.

→ Importance :-
- The "heart" of the architecture, ensuring smooth operations between client & database.
- Facilitates addition, deletion, or updating of data.

3) Database Tier

→ Role :- Bottommost tier ; manages data storage

→ components :-

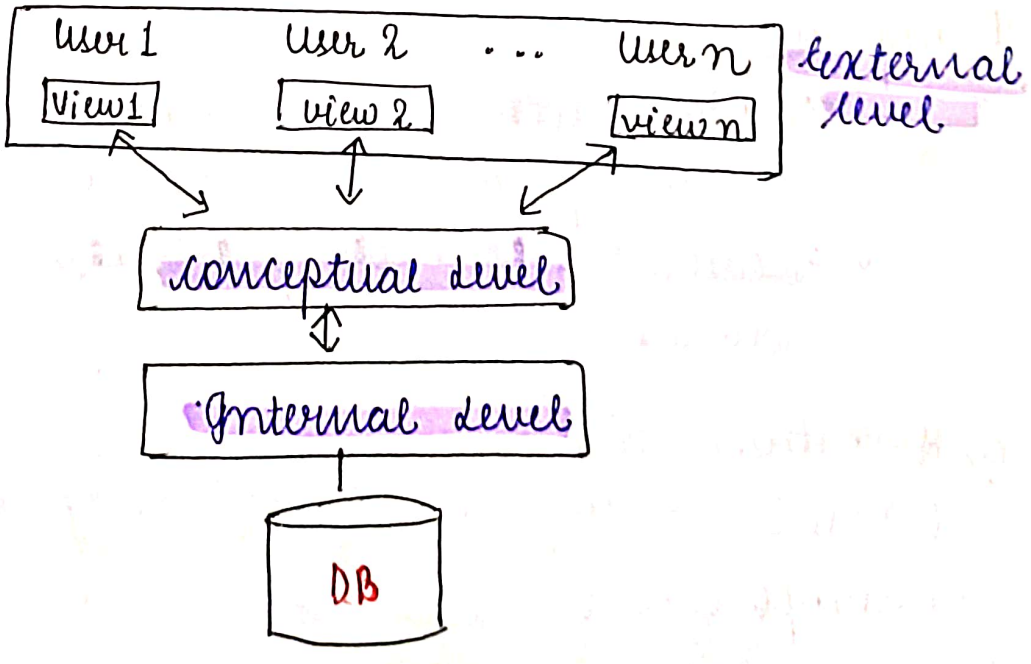
- RDBMS :- MySQL, PostgreSQL, etc.
- DBMS :- MongoDB, CouchDB, etc.

→ Responsibilities :-
- store and retrive data for application tier.
- Manages database security and consistency.

→ Interaction
Directly interacts with the application tier, not the client tier.

- Advantages of three-tier Architecture.
1) Independent Development - Each tier can be developed and deployed separately.
2) Scalability - Tiers can be scaled independently based on load.
3) Improved security - Separation ensures restricted access between tiers.
4) Flexibility - Easier to update and maintain individual tiers.
5) Reliability - Robust due to the logical and physical separation.

⇝ Generalized DBMS three-tier Architecture



Views of Data

- The primary goal of database is to store and retrive data efficiently while managing complexity.
- **Data Abstraction** - Hides complexity by providing only essential details.

Levels of Data Abstraction

There are 3 levels of abstraction in database.

1) **Physical Level (Internal level)** - How data are stored.
   Lowest level; deals with the physical storage of data on hardware (eg HDDs, SSDs).
   Focus: How data is stored (eg data structure like arrays, linked list).
   Features:
   - Use complex data structures for efficient storage.
   - Handles multimedia and other complex data types.
   - Ensure physical data independence: changes in physical storage do not affect the logical structure.

2) **Logical level (Conceptual level)** - what data are stored
   Middle level; defines what data is stored and the relationship among data.
   Focus: The structure of database (Schema)
   Features:
   - Provide logical data independence: changes in the logical structure do not affect the application layer.
   - Used by database administrators to define the database schema.

▷ Eg ER models, relational data models.

3) View level (external level)

Topmost level; interacts with users.

Focus: Hides complexity and provides a user-friendly interface.

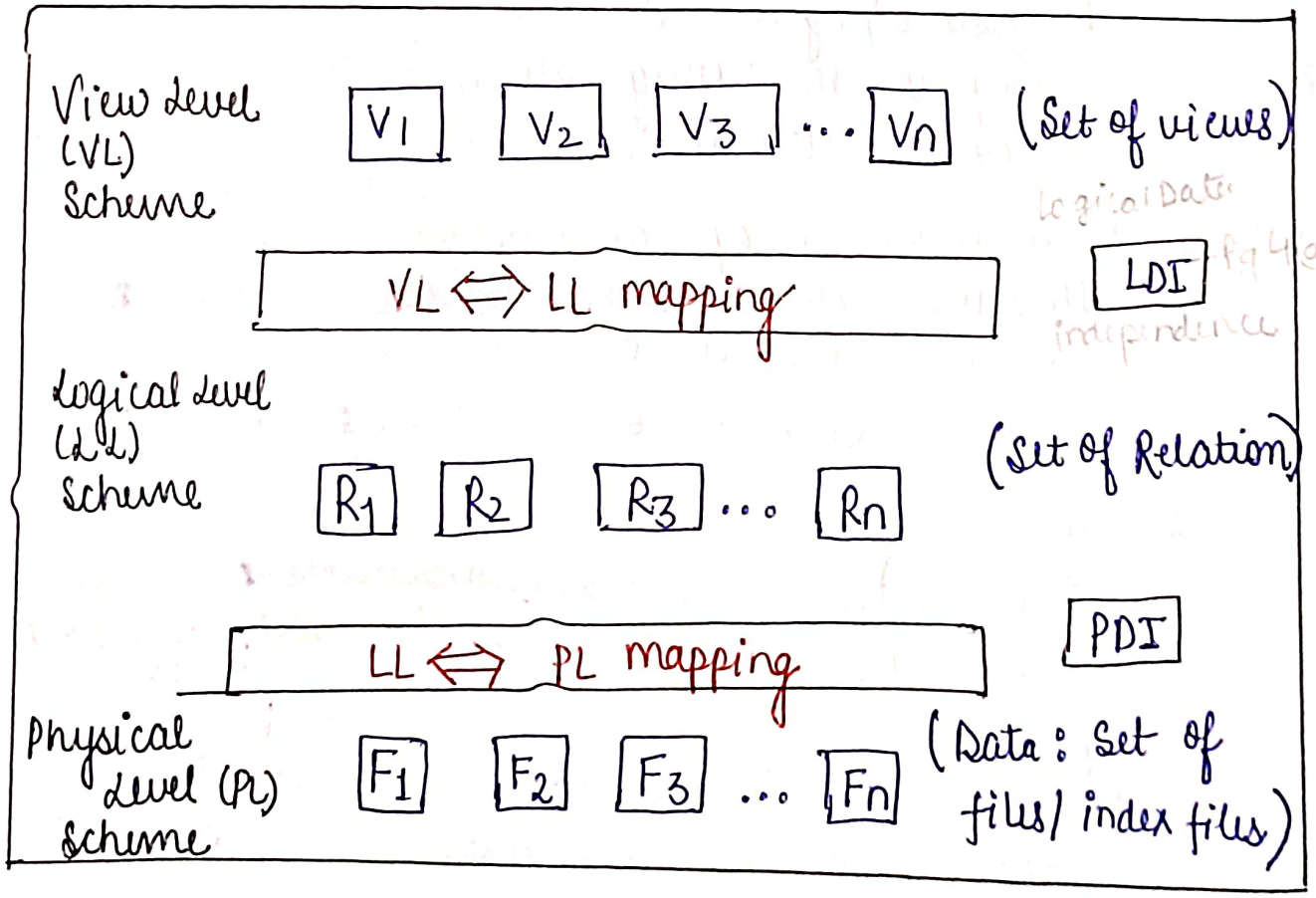
Features:

- ▷ Provide multiple views for different users based on their roles.
- ▷ Handles security by restricting access to sensitive data.

Advantages of data Abstraction :-

1) Simplifies database design and maintenance.
2) Provides flexibility through data independence.
3) Enhance security with multiple views.
4) Improves user experience by hiding underlying complexities.

Abstraction levels in DBMS



• **Physical Data Independence :-** (PDI)
The ability to modify physical level schema without affecting the logical or view level schema.

→ **Performance tuning :-** modification at physical level, creating a new index, etc.

Physical Data Independence - modification is localized
- achieved by suitably modifying PL-LL (Physical level - logical level) mapping.
- A very important features of modern DBMS.

• **Logical Data Independence :-** (LDI)
The ability to change the logical level schema without affecting the view level schemas or application program.

→ Adding a new attribute to some relation
- no need to change the programs or views that don't require to use the new attribute.

→ Deleting an attribute
- no need to change the programs or views that use the remaining data.
- view definition in VL-LL (view level - logical level) mapping only need to be changed for views that use the deleted attribute.

• Mapping
The process of transforming requests and result between the levels are called mappings.

Development process of Database system

Step 1 Requirement collection

- Data model requirement
  - Various pieces of data to be stored and the interrelationships.
  - presented using a conceptual data model such as E/R model
- Functional requirement
  - Various operations that need to be performed as part of running the enterprise.
    Eg acquiring a new book, enrolling a new user, issuing a book to user, etc.

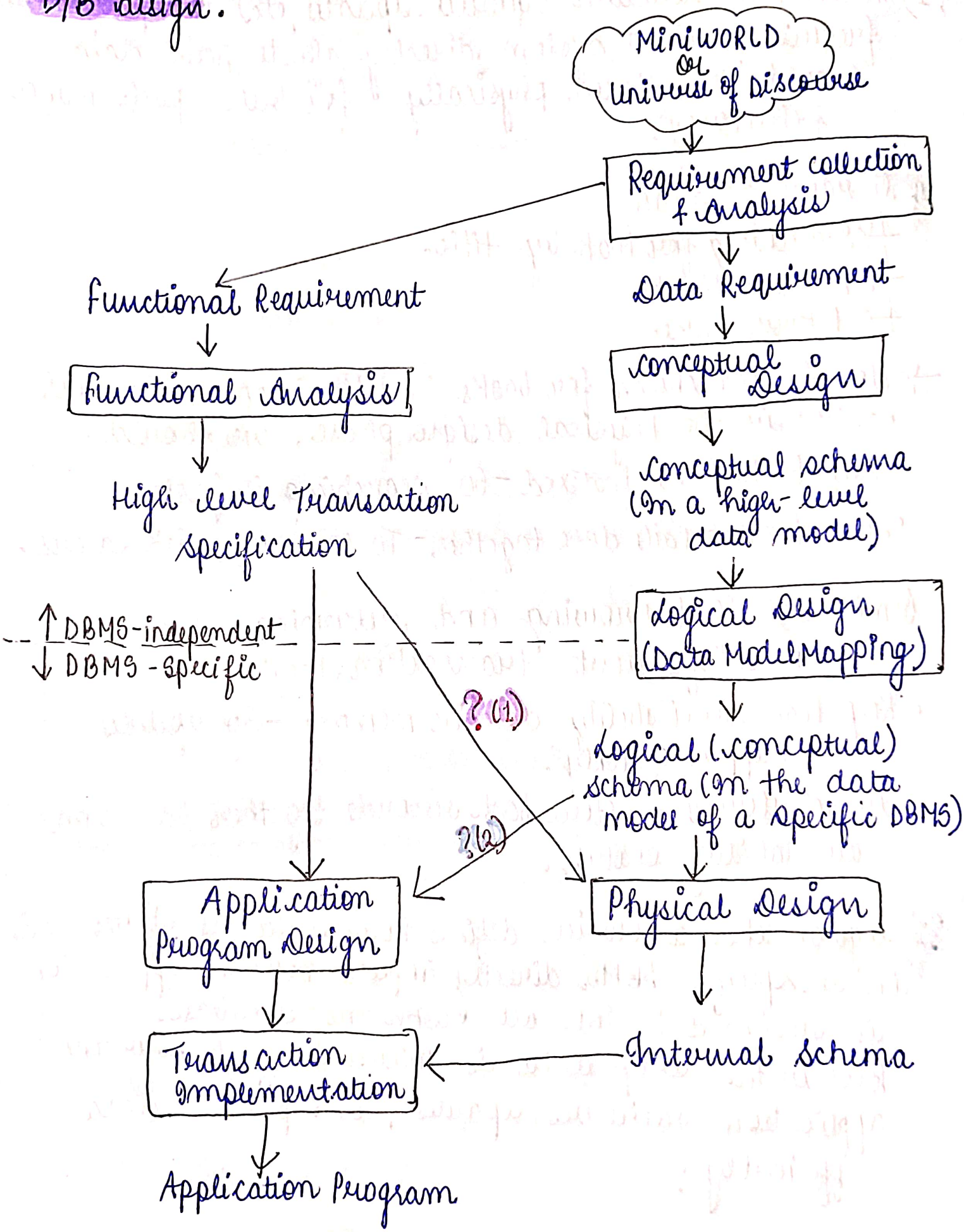
Step 2: Convert the data model into a representation level model

- typically relational data model.
- choose a RDBMS system & create the database.

Step 3: Convert the functional requirement to application programs

- Programs in a High-level language that use embedded SQL to interact with the database and carry out the required task.

D/B design.


MiniWORLD
or
Universe of Discourse
Requirement collection & Analysis
Functional Requirement
Data Requirement
Functional Analysis
conceptual Design
High level Transaction Specification
Conceptual schema (In a high-level data model)
↑ DBMS-independent
↓ DBMS-specific
Logical Design (Data Model Mapping)
? (1)
Logical (conceptual) schema (in the data model of a specific DBMS)
? (2)
Application Program Design
Physical Design
Transaction Implementation
Internal Schema
Application Program

?1) The way transactions (queries, updates, etc) are performed frequently in a system directly affects how data should be stored physically for better performance & efficiency.

Eg: library database
- Searching for book by title.
- Borrow books
- Return books

→ Now if searching for books by title happens the most, then in the physical design phase, we should.
- Keep book titles indexed - So searching is fast
- Store book details close together - To minimize disk access.

Similarly, if borrowing and returning books are the most frequent transaction, then:
- Keep book availability data in memory - So updates happen quickly.
- Use clustering - (Store book records together based on availability status).

?2) Logical schema (which defines how data is structured in a specific DBMS) directly impact how the application is designed to interact with the database. Because the way data is modified affect how the application retrieves, updates, and process data efficiently!

• Data Model
collection of conceptual tools to describe the database at a certain level of abstraction.

⇒ "Data + Relationship + Semantics + Constraints"

→ Conceptual Data Model eg E-R Model
- a high level description
- useful for requirement understanding.

→ Representational Data Model eg Relational Model
- describing the logical representation of data without giving details of physical representation.

→ Physical Data Model
- description giving details about record formats, file structure, etc.

• Data vs Schema or Meta-Data
↳ DBMS is generic in nature
  ↳ not tied to a single database
  ↳ capable of managing several database at a time.

↳ Data & schema are stored separately.

→ Data definition = setting up skeleton structure

→ Database loading / population = storing data.

Database Users :-

People who work with database can be categorized as database users or database administrators.

1) Naive Users

User with no technical knowledge of database but can interact with the system through predefined application.

eg • ATM user withdrawing money.
• Booking tickets online.
• Bank clerks/tellers accessing customer details.

★ Do not need to write SQL queries or understand dB structure.
★ Interact via - user friendly interface like web or mobile.
★ Unsophisticated users.

2) Application Programmed.

Professionals who design and develop user interface and application programs used by naive users.

Role :- 1) Write software using programming language & tools (RAD).
2) Create user friendly application to make dB interactions seamless.

eg software developers building ticket booking system, mobile apps, etc.

3) Sophisticated Users

Users who directly interacts with the database using query languages and data analysis tools.

eg Database Administrators: Write SQL queries to fetch or manage data.
Data Analysts: Use data analysis software to extract insights.

★ Understand database structures and query language.
★ Perform tasks like data fetching, analysis and report generation.

4) Specialized Users

Advanced user with expertise in handling specialized database application.
(or)
Sophisticated users who write specialized database application.

Role: 1) Work with complex systems like multimedia dB, knowledge bases, AI-driven system.
2) Proficient in maaging & designing next generation dB application.

eg • Experts in CAD
• Developers handling multimedia data (eg audio, video, animation, etc).

• Database Administrator (DBA)

A person who has such central control (control on data and programs that access those data) over the system is called Database Administrator.

Function of DBA :-

a) Schema: Overall design of the database, including tables and columns.
- The DBA uses Data definition language (DDL)
  - ↳ • Define the database structure.
  - ↳ • Creates tables and their storage structures.
  - ↳ • Decide the access methods for the data.

b) Storage and Access management:
- Determines the storage structure and method to access data.
- Ensure the storage is efficient and data access is authorized.

c) Schema and Physical Organization Modification :
- Schema modification: updates the dB overall design as needed by the organization.
- Physical organization Modification : Adjust data storage and organization at the physical level. (to improve performance)

d) Authorization of Data Access :
- Grant privileges to users based on their roles.
- Defines what data each users can access to maintain security.
- The autorization information is kept in a special system structure that the dB system consults whenever someone attempts to access the data in the system.

e) Routine Maintenance Activities:-

- Periodic Backups:
  → Essential for data recovery in case of hardware/software failures or natural disasters.
  → Ensure a backup copy is maintained & stored in remotely for added safety.

Disk Space Management:
→ Monitors available disk space to accommodate data growth (eg new customer records).

Performance Optimization:
→ Ensure database respond to user queries quickly.
→ Periodically checks and optimizes database performance.

⇒ These responsibilities ensure that the database remains functional, secure and efficients for organization needs.

- History of DBMS

→ 1950s - Early 1960s: Magnetic Tapes
- Magnetic tapes were used for data storage.
- Data access was sequential, not random.

eg Payroll processing using magnetic tapes.

- Challenges:-
  → Sequential data access made direct access impossible.
  → Data size often exceeded primary memory, requiring merging of data.
  → Complexities in processing due to sequential nature.

→ Late 1960s - Early 1970s: Hard Disk Drives (HDD)

- Hard disk replaced magnetic tapes.
- Enabled direct/random data access, eliminating sequential limitation.
- Relational model introduced by Edgar Frank Codd:
  - → Defined non-procedural querying.
  - → Introduced data-abstraction.
  - → Earned the ACM Turing Award for contribution.

→ 1980s: Commercial Relational Database

- Relational databases became commercially viable after overcoming initial drawbacks.
- Replaced network and hierarchical models:
  - → Network/Hierarchical models were tightly coupled with implementation, making changes complex.
- Research began on
  - → Parallel database
  - → Distributed database
  - → Object-oriented database.

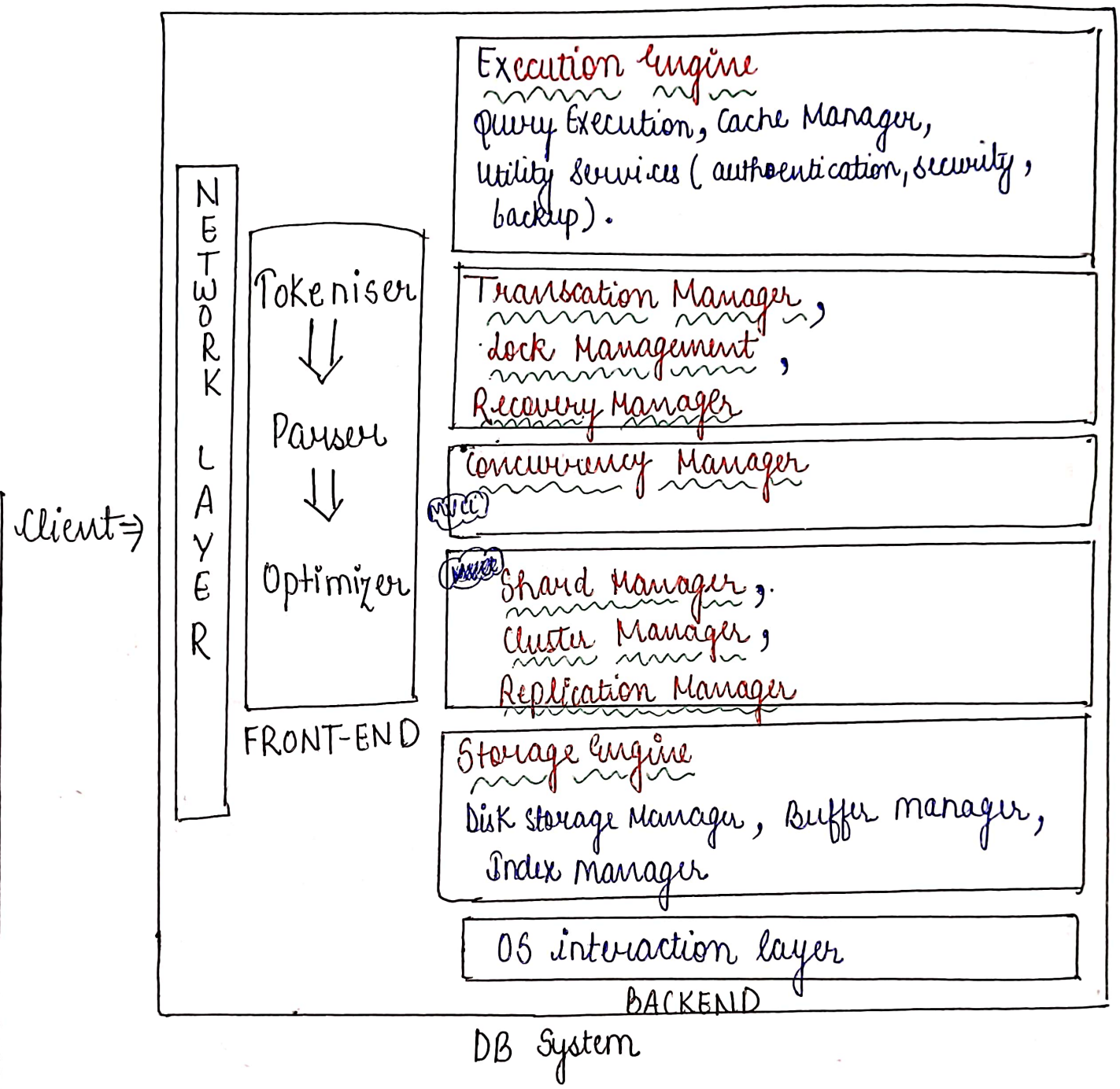
→ 1990s: Growth of SQL and the Internet

- SQL (Structured Query Language) used for decision support application.
- Introduction of
  - → Parallel database products.
  - → Object-relational support.
- Growth of the Internet and WWW
  - → Database supported high transaction rates.
  - → Emphasis on 24x7 availability and high reliability.
  - → Minimal or no downtime for maintenance.

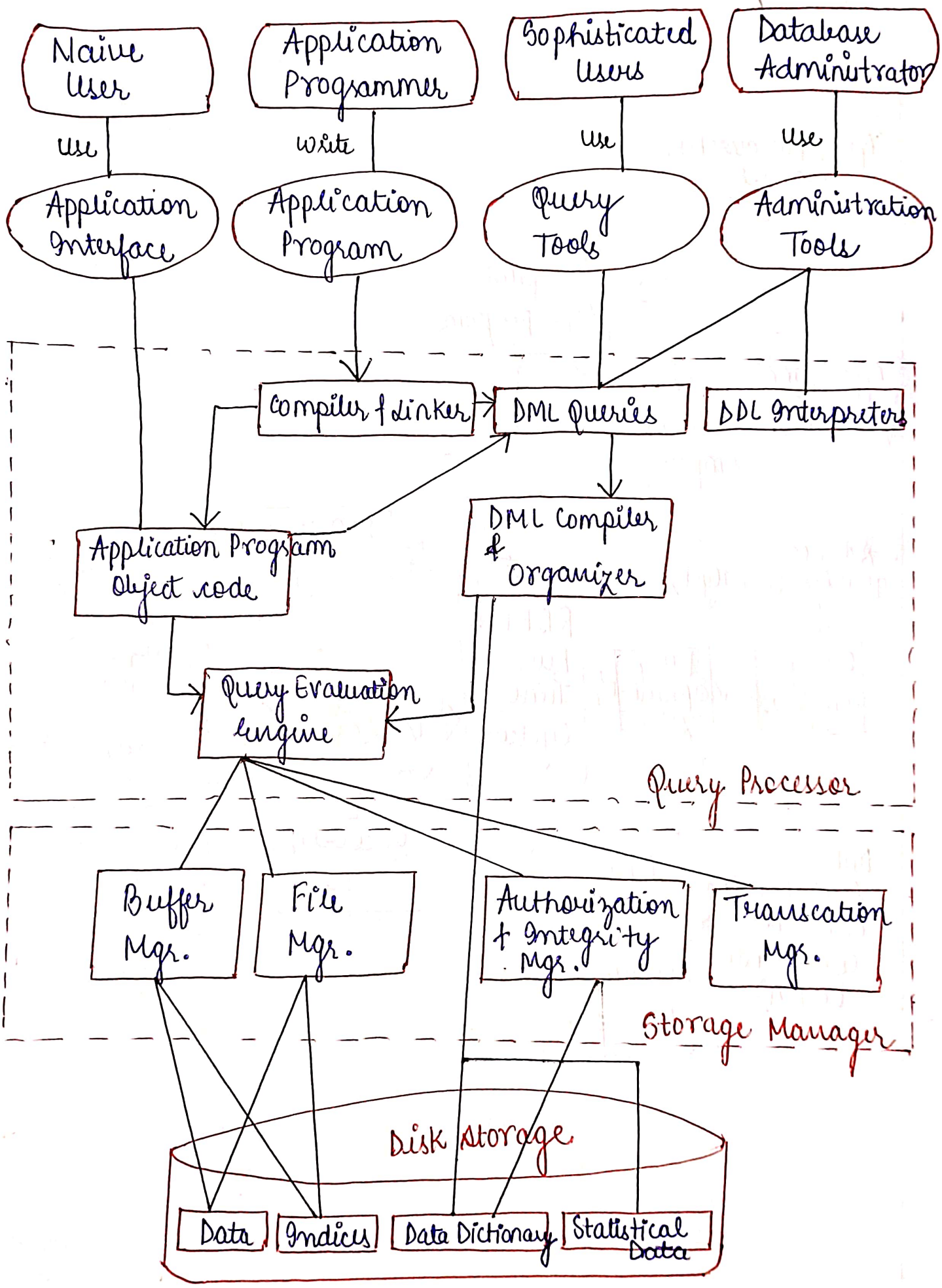
2000s: Emerging Technologies

- XML (Extensible Markup Language)
  - Used for data exchange and storing complex data type.
- XQuery introduced as an XML query language.
- Development of novel distributed storage system for large-scale website (eg Amazon, Google, facebook).
- Introduced "autonomic-computing / auto-admin" features to simplify database management.

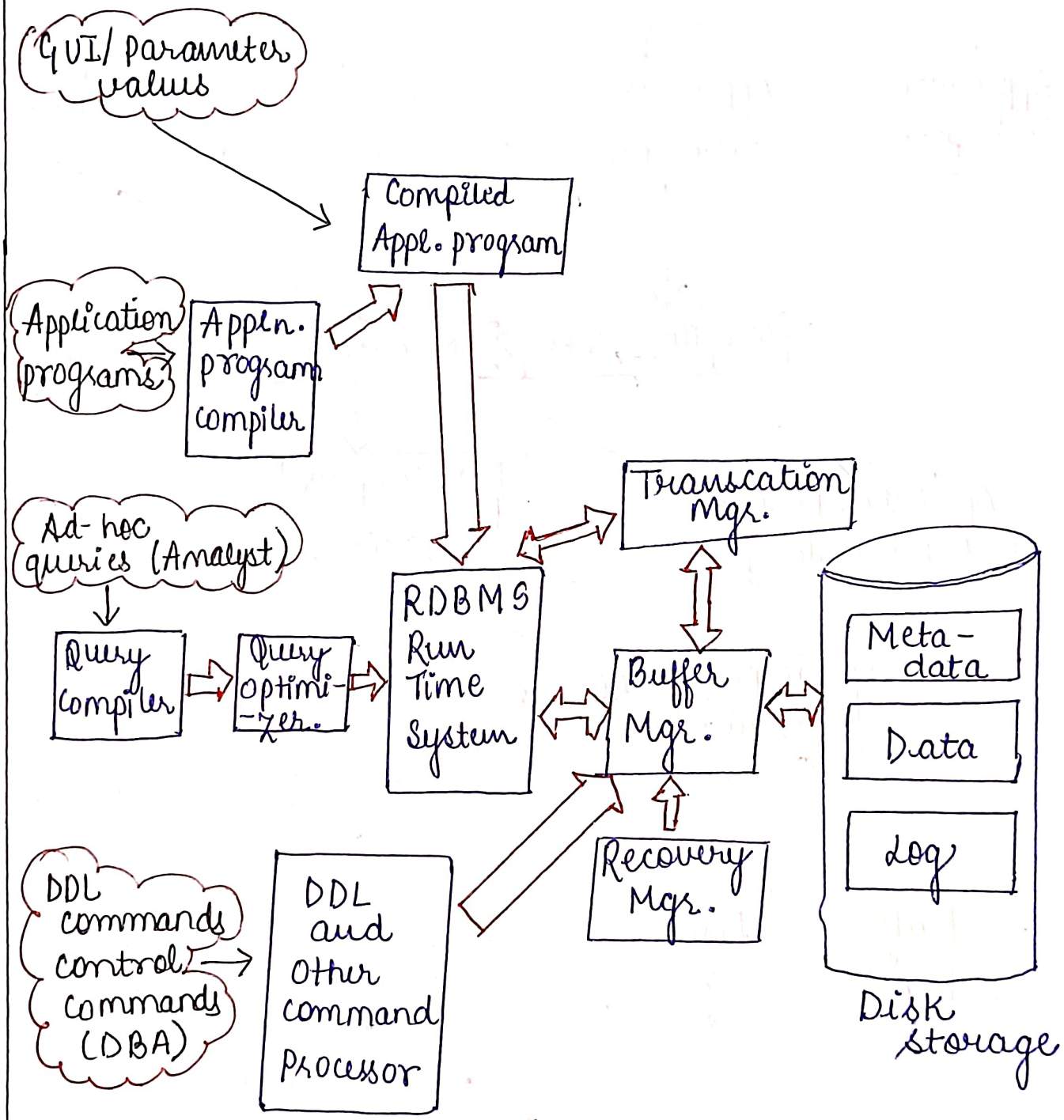
Database Architecture (General)



DBMS Architecture



RDBMS Architecture



Storage Manager

- Acts as a bridge / interface between application programs (top-level) and data stored in disk storage (low-level).
- Handles queries submitted to the system via the query processor.
- Manages interaction with the file system used by the operating system.
- Converts DML (Data Manipulation language) commands into low-level file system commands.

Thus, the storage manager is responsible for storing, retrieving and updating data in the database.

Components of the storage Mgr. :-

1) Buffer Mgr -
- Fetch data from disk storage to main memory.
- Decides what data to cache in memory for faster access.
- Handles large database sizes efficiently.

2) File Manager -
- Allocates space on disk storage.
- Manages data structures used for storing information.

3) Authorization & Integrity mgr -
- Enforces integrity constraints (eg balance ≥ 0)
- Manages concurrent transactions without conflicts.

4) Transaction mgr -

- Database management system are crucial due to the enormous volume of data they handle.
- Data volume extends from Kilobytes to yottabytes and beyond.

Challenges :-

1) Main memory is volatile and cannot store database.
2) Hard disk and SSD may not be sufficient for large-scale data.

Solution :-

Storage servers → data centres → cloud storage

Eg:- Facebook and WhatsApp handles enormous data volumes daily.
Deleted data is still stored in servers, making data management complex.

[DBMS functionality] → {Storage mgr.}
[DBMS functionality] → {Query processor}

• Ensure data consistency despite system failure. (BC)
• Manages concurrent transactions without conflicts.

Data storage :-

The storage mgr. implements several data structures as part of the physical system implementation:

1) Data files - stores the actual database.
2) Data dictionary - stores meta-data.
3) Indices -
   • Provides fast access to data.
   • Work like an index page in a book to locate specific data efficiently.
4) Statistical Data - Stores statistics about data to assist in database management decisions.

Query Processor

Handles queries from users or applications and interacts with the database to execute them.

Components of the query processor:-

1) DDL interpreter
2) DML compiler
3) Query evaluation engine

Users Interacting with the Database:-

- Database Administrator (DBAs) - Use DDL and DML commands to manage databases.
- Sophisticated User/Analysts - Use query tools to execute DML queries.
- Application programmers - Write programs that send queries to the database.
- Naive Users - Use application interfaces to interact with the database indirectly.

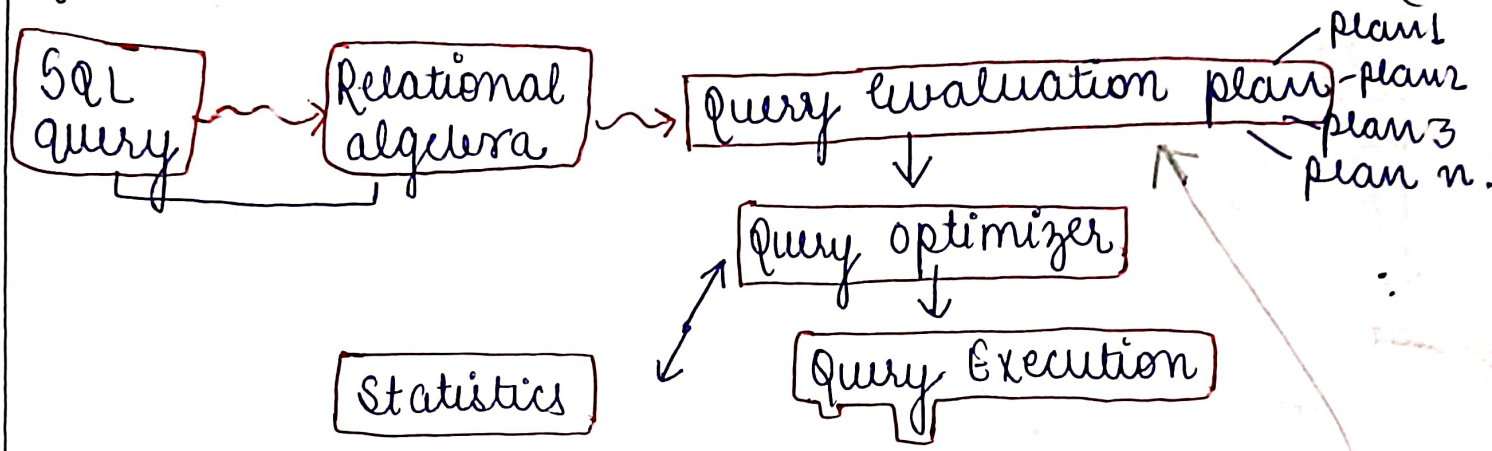
1) DDL Interpreter
- Interprets DDL (Data definition language) statements.
- Stores schema definition in the Data dictionary (metadata storage).

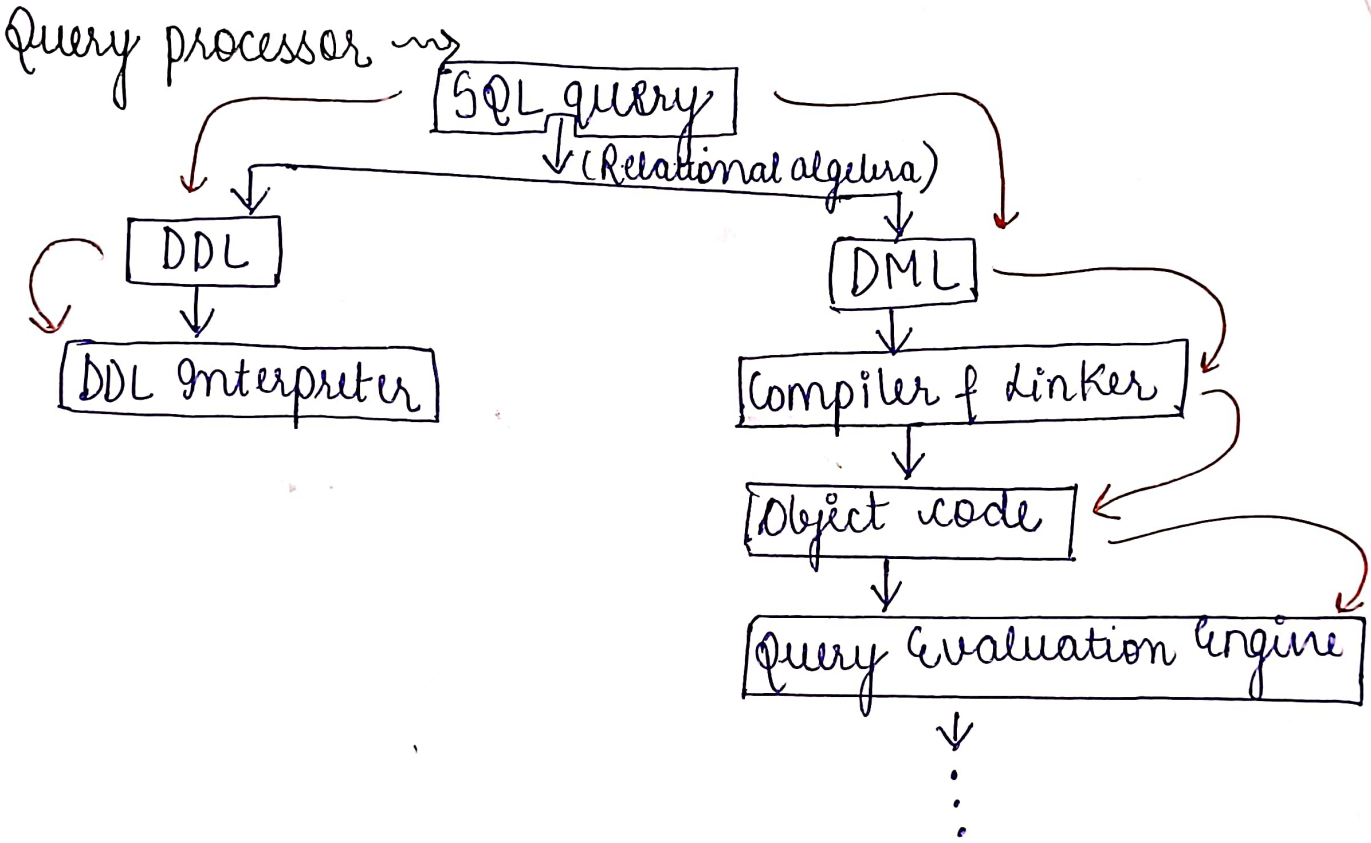
2) DML Compiler
- Translate DML (Data Manipulation language) queries into query evaluation plans.
- Optimizes queries by selecting the lowest cost execution plan.

3) Query evaluation Engine (QEE)
- Executes query evaluation plan on the database.
- Chose the best execution plan based on cost.

QEE –

(6d)


SQL query
Relational algebra
Query evaluation plan
plan1
plan2
plan3
plan n.
Query optimizer
Statistics
Query Execution


Query processor
SQL query
(Relational algebra)
DDL
DML
DDL Interpreter
Compiler & Linker
Object code
Query Evaluation Engine