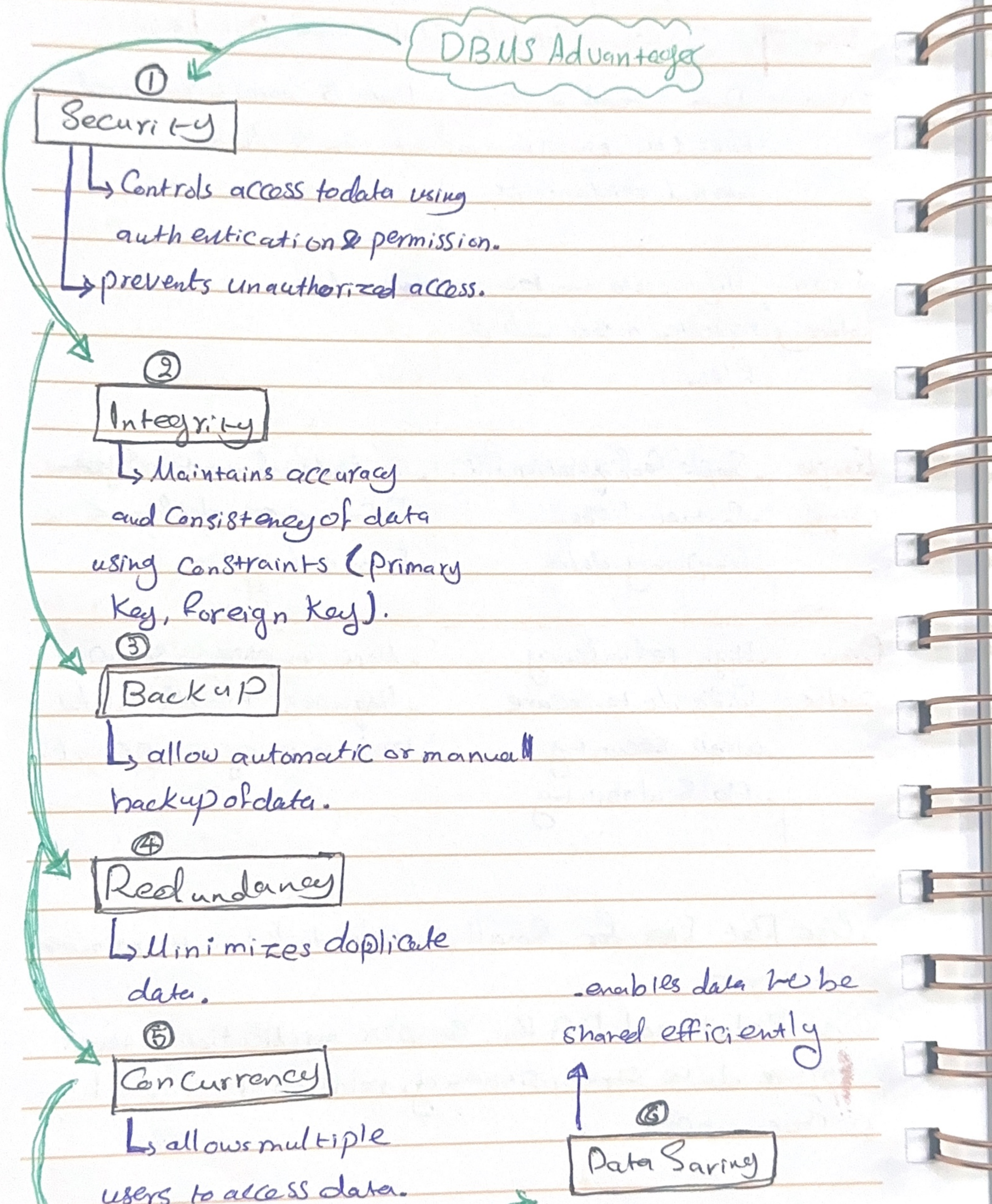
# 1) Comparison Assignment

| | ~~Flat file~~ System / Flat file | Relational Databases |
|---|---|---|
| Structure | - Data is stored in plain files, (txt, csv..) with no internal relationships | - Data is stored in structured with row & Colums |
| Data Redundancy | - High-data may toe repeated across multiple files. | Low - data |
| Example Usage | - Simple Configuration file.<br>- Contact lists.<br>- Temporary data | - Student information Systems<br>- E. Commerce platforms.<br>- Financial system. |
| Draw Backs | - High redundency.<br>- Difficult to secure<br>- Weak security.<br>- No Scalability. | - More Complex to set-up.<br>- Requires DBMS Soft/w<br>- Needs ongoing mangement. |

- Use Flat files for small, simple tasks with minimal structure.
- Use Relational DB for complex applications that require data consistency, relationships and performance.

# 2) DBMS Advantage Mind Map :



## 5) Roles in a DB systems.

① System Analyst → - Identifies system requirements from users.
→ Analyzes current processes and proposes improvements.

② DB Designer → Designs the structure of DB. (ERD)
→ Defines tables, relationships, primary and foreign keys.

③ DB Developer → Buids the database using SQL.
→ Writes stored procedures, indexes, triggers

④ DB Administrator → Manage the performance and Security of database.
→ Handles backup, recovery, and user permissions.

⑤ Application Developer
→ Develops the application or front-end that interacts with the database.
→ Uses SQL or API to retrieve and manipulate data.

⑥ BI Developer (Business Intelligence)
→ Uses data analysis tools (Power BI)
→ Creates reports and deshboards to support decision-making.

## ④ Type of DB ...

### 1) Relational Database (RDBMS) =

⇒ Stores data in tables with rows & columns.
⇒ Supports SQL & relationships (MySQL, Oracle)..
⇒ Best for structured data with clear relationships.

### 2) No SQL Database =

⇒ Designed for unstructured or semi_structured data.
⇒ Types include ⇒ Document (MongoDB), Key-Value (Redis), Column (Cassandra), Graph (Neo4j).
⇒ Scalable and flexible, used in big data and real-time applications.
⇒

### 3) Cloud Database =

⇒ Hosted on cloud platforms (AWS RDS, Google Cloud SQL, Azure Cosmos DB).
⇒ Accessible via the internet, Scalable, and managed by providers.

### 4) object-Oriented DB =

⇒ Stores data as objects, Like in object-oriented Programming (OOP).
⇒ Suitable for Applications built in OOP Languages (C++, Java)..

### 5) Distributed DB. =

⇒ Data is Stored across multiple physical Locations, often Connected via network.

⇒ Ensures high availability and performance ( Google Spanner).

### 6) Graph Database. =

⇒ Focuses on relationships between data using nodes and edges ( Neo4j).

⇒ Ideal for Social Networks, recommendation engines.

### 7) Time-Series DB. =

⇒ Optimized for Storing time-stamped (temperature, stock prices).

### 8) Hierarchical Database =

⇒ Data is organized in a tree-like structure ( parent-child relationships)

⇒ Example: IBM's, IMS.

(DB) / (use Cases).

1) Relational ⟹ Banking System, School DB.

2) Non Relational ⟹ IOT apps, real time analystics.

3) Centralized ⟹ Small businesses, Local Applications.

4) Distributed ⟹ Global apps, real time system.

5) Cloud ⟹ Web apps, SaaS, Mobile backends, start-up.

# 5) Cloud Store & DB..

⇒ What is Cloud Storage? is Service that allows users and organizations to store data on remote servers, they can hosted by cloud providers and access it by internet.

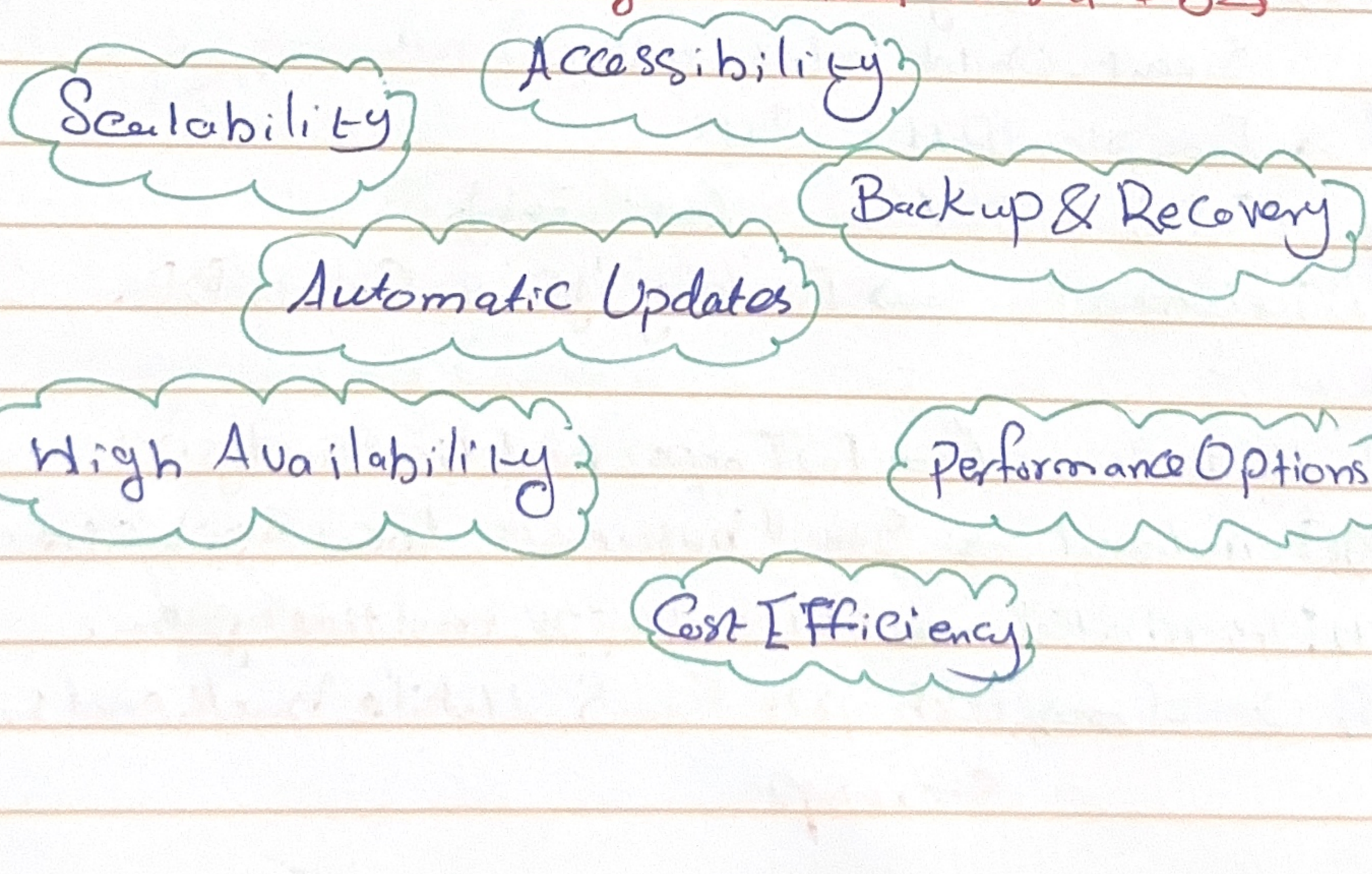
Example :- Amazon S3, Google Cloud Storage, Microsoft Azure Blob Storage.

⇒ How Cloud Storage Relates to DB?

- Cloud Storage is general-purpose, but cloud DB are specialized tools built on top of cloud infrastructure to store, manage and query structured or unstructured data.
- Cloud database run on virtual cloud servers and often use Cloud Storage to persist data.

Examples :- Azure SQL / Amazon RDS / Google Cloud Spanner.

⇒ Advantage of Cloud-Based DB ⇒



## • Disadvantage of Cloud-Based DB ⇒

- Security Concerns
- Internet Dependency
- Vendor Lock-in
- Latency
- Limited Control

## • What is DB Engine ?

- DB engine or ( DB management System engine) is the Core S/W Component that is
  1) Stores, retrieves, manages and processes data.
  2) Excutes SQL queries and handles tasks like transactions, indexing and Concurrency.

[ It is the brain of the DB system ] ...

| DB Engine | Language Used | Notes |
|---|---|---|
| 1) SQL Server | T-SQL (Transact-SQL) | - Microsofts extension of SQL with procedural featu |
| 2) Oracle | PL-SQL | - Oracle procedural extension SQl Support advanced programming; ~~with its to own Syntax and extension~~ |
| 3) My SQL | ANSI SQL + My SQL | Uses Standard SQl with SQL its Own syntax and extensions. |
| 4) Posrger SQL | LP / PgSQl + ANSI SQL | Supports SQl and procedural language Similar to Oracles PL/SQl |

⇒ Is there a Relationship between Engine and Language ??

→ Each DB engine typically comes with it's own SQl dialect or extension :.

- These extend standard SQl (called ANSI SQL) with engine-specific features, such as error handling, loops and functions.
- For Example as PL/SQL only works with Oracle, and T-SQL is designed for SQl Server.

⇒ Can one language work across different engines?

Yes and No

- Standard SQL (ANSI SQL) ⇒ work across most engines (ex: SELECT, INSERT, WHERE).
- Procedural extensions ⇒ Engine-Specific. Example (T-SQL, PL/SQL, PI/PgSQl) won't run on other engines.
- Portability requires caution → when building cross-engine apps, avoid using engine specific syntax or features.

- DB Engine ⇒ Executes and manages data regeries
- Language ⇒ Most engines use SQL, but extend it with their own features.
- Cross-Compatibility → basic SQl is protable, but advanced scripts usually aren't.

# Can We Transfar a DB between Engines ?

Yes, but it requires Carful planning and tools.

- It is possible to migrate a DB betweens engines (from SQl server to MySQl or Oracle to Postgre SQL), but it requires Carfull planning and tools due to key differences between systems.

## Challenges of Engine-to Engine Migration :-

A) SQL Dialect Differences ⇒ each engin have its own SQl flavo
B) Data Type Incompatibility ⇒ Some data type don't match exact
C) Stored procedures & Triggers ⇒ Procedural languages aren't portab
D) Functions & Views ⇒ Need to manually reviewed or updated
E) Performance Tuning Differences ⇒ Indexes, optimization
F) Permissions & Roles ⇒ Security must be reconfigured.

## What Should we Consider Before Transferring ??

Data Types / Tiggers Procedures / Constraints Keys
Encoding && Collation / Indexes / Tools
pgloader / Oracle SQl developer

Examples Tools for migrations

| Source | Target | Recommended Tool |
|---|---|---|
| SQL Server | My SQL | My SQL Workbench |
| Oracle | PostgreSQL | Ora2Pg |
| SQL Server | PostgreSQL | SQLines |

⇒ What is a logical Scherma?

A) Abstract design of a database

B) It defines what data is stored and how the entites are related, but how it is stored physically.

c) it focuses on tables, Columns, data types, relationships, Constraints.

⇒ what is a physical Scherma?

A) defines the actual implementation

B) It includes storage formats, indexes, partitions and access paths.

C) It's platform-specific and optimized for performance.

⇒ ~~What is~~ Difference Between logical & physical?

| Logical | physical |
|---|---|
| - What data is stored and its structure. | - how data is stored and accessed. |
| - High (Conceptual view) | - low |
| - platform-independent | - platform-specific |
| - Tables, relationships, Constraints. | - Indexes, file structure, partitions. |
| - DB designers, analysts | - DBAs, System architects. |

- Why Is It important to Understand Both?

⇒ Logical schema help in modeling & understanding the data structure.

⇒ Physical schema is essential for performance tuning & efficient storage.

⇒ Together, they ensure a well-designed, scalable and efficient DB system.

---

Example: 'Student" Entity..

→ logical Schema:

Tables: Student

- StudentID
- First-Name
- Last Name
- Email
- DateOfBirth

Relationships:
Student may be linked to Course table via Enrollment.

Physical Schema:

```
CAEATE TABLE student(
student_id SERIAL PRIMARY Key,
first_name VARCHAR (50),
last_name VARCHAR (50),
email #VARCHAR (100) UNIQUE,
date_of_Birth DATE.
)TABLESPACE Studentspace;
CREATE INDEX idx_email No Student(Email);
```