Types of Databases

Relational vs Non-Relational

- **Relational Databases** (e.g., SQL Server, MySQL, PostgreSQL)
  - Organize data into tables with rows and columns.
  - Use Structured Query Language (SQL) for querying.
  - Ensure data integrity through ACID compliance.
- **Non-Relational Databases** (e.g., MongoDB, Cassandra)
  - Store data in flexible formats like JSON (MongoDB) or wide-columns (Cassandra).
  - Good for handling unstructured or semi-structured data.
  - Often scale horizontally and prioritize performance and scalability over strict consistency.

Centralized vs Distributed vs Cloud Databases

- **Centralized Database**: All data stored in a single location.
  - Easy to manage but a single point of failure.
- **Distributed Database**: Data is stored across multiple physical locations.
  - Offers high availability and fault tolerance.
  - Requires complex synchronization.
- **Cloud Database**: Hosted on cloud platforms (e.g., AWS, Azure, GCP).
  - Scalable and managed infrastructure.
  - Can be relational (Amazon RDS) or non-relational (Firebase).

Use Case Examples

- **Relational**: Banking systems, ERP systems.
- **Non-Relational**: Social media feeds, IoT data.
- **Cloud**: E-commerce platforms needing global scalability.

Cloud Storage and Databases

What is Cloud Storage?

- A service where data is remotely maintained, managed, and backed up over the internet.
- Related to databases as many cloud database solutions rely on cloud storage for data persistence.

Advantages of Cloud Databases

- **Scalability**: Automatically adjusts resources based on demand.
- **Cost-effective**: Pay-as-you-go pricing.
- **Managed Services**: Automatic backups, patching, and maintenance.

Disadvantages

- **Latency**: Network dependence can affect performance.
- **Security/Compliance**: Data privacy concerns.
- **Vendor Lock-In**: Hard to migrate away from a cloud provider.

Database Engines and Languages

What is a Database Engine?

- The core software component that manages data storage, retrieval, and modification.
- Responsible for query processing, indexing, transactions, and concurrency.

Examples and Their Languages

- **SQL Server**: Uses **T-SQL**
- **MySQL**: Uses **ANSI SQL** with some custom extensions
- **Oracle**: Uses **PL/SQL**
- **PostgreSQL**: Uses **PL/pgSQL** and supports **ANSI SQL**

Engine-Language Relationship

- The engine determines the dialect of SQL supported.
- Some languages (like ANSI SQL) are standardized and work across multiple engines with minor differences.

Cross-Engine Language Use

- **Basic SQL** can often be used across engines.
- **Advanced features** (like stored procedures or custom functions) may not be portable without rewriting.

Can We Transfer a Database Between Engines?

Is Migration Possible?

- Yes, you can migrate between engines (e.g., SQL Server to MySQL, Oracle to PostgreSQL).

Challenges

- **Data type incompatibility**
- **Different SQL dialects**
- **Stored procedures and triggers need rewriting**
- **Performance tuning differences**

Pre-Migration Considerations

- **Schema compatibility** (data types, keys, constraints)
- **Dependencies** (views, functions, procedures)
- **Data integrity**
- **Testing and validation**

Logical vs Physical Schema

Logical Schema

- Describes *what* data is stored and *how* it's related.
- High-level design (e.g., entities, attributes, relationships).

Physical Schema

- Describes *how* data is actually stored in the database.
- Includes tables, columns, data types, indexes, and storage structures.

Differences

Logical Schema	Physical Schema
Conceptual design	Actual implementation
Independent of DBMS	DBMS-specific
Focuses on structure	Focuses on performance and storage

Why Both Matter

- Logical schema ensures business requirements are met.
- Physical schema ensures performance and storage efficiency.