Database Basics

What is a Database?

A database is an organized collection of data that is stored and managed electronically. It allows for efficient storage, retrieval, and management of data. Databases are used in a wide range of applications, from websites and mobile apps to enterprise software and analytics.

Example:

Imagine a library where all books are stored randomly on shelves without any order. Finding a book would be difficult. Now, if the books are organized by genre, author, and title, it becomes easy to locate any book. A database works similarly by organizing data in a structured way for easy access.

Types of Databases

1. SQL (Relational) Databases

- **Structured Query Language (SQL)** is used to manage and query the data.
- Data is stored in **tables** with rows and columns.
- **Schema-based**: The structure of data is predefined, and relationships between tables are established using **foreign keys**.
- Examples: MySQL, PostgreSQL, Oracle, SQL Server.

Scenario:

A university database stores student data, course information, and enrollment records. Each type of data is stored in separate tables, and relationships between them (e.g., which student is enrolled in which course) are managed using foreign keys.

Example Query:

```
SELECT student_name, course_name FROM students

JOIN enrollments ON students.student_id = enrollments.student_id

JOIN courses ON enrollments.course_id = courses.course_id;
```

2. NoSQL (Non-Relational) Databases

- Data is stored in various formats: **documents**, **key-value pairs**, **wide-column stores**, or **graphs**.
- **Schema-less**: The structure of data is flexible, suitable for unstructured and semi-structured data.
- Examples: MongoDB (Document), Redis (Key-Value), Cassandra (Wide-Column), Neo4j (Graph).

Scenario:

An e-commerce website needs to store product information, including varied attributes like size, color, and specifications. Using a document-based NoSQL database like MongoDB allows flexible storage of product data as JSON-like documents without a rigid schema.

Example Document (in MongoDB):

```
{
  "product_name": "Laptop",
  "brand": "Dell",
  "specs": {
    "RAM": "16GB",
    "Storage": "512GB SSD"
  }
}
```

Advantages of Relational Databases (SQL)

1. **Structured Data Management:** Best for applications with well-defined data structures.
2. **Data Integrity:** Enforces data consistency through constraints and relationships.
3. **ACID Compliance:** Guarantees **Atomicity**, **Consistency**, **Isolation**, and **Durability** of transactions.
4. **Complex Queries:** Supports advanced queries using SQL.
5. **Security:** Provides robust access controls and data security.

Example Scenario:

A banking application where transactional integrity and data consistency are crucial. SQL databases can handle complex queries, ensure data consistency, and maintain the accuracy of financial transactions.

Key Takeaway:

- Use **SQL databases** when data is structured, and relationships are important (e.g., banking, e-commerce).
- Use **NoSQL databases** when data is unstructured, requires scalability, or involves varied data formats (e.g., social media, real-time analytics).