HEXAWARE TRAINING

STRUCTURED QUERY LANGUAGE (SQL)

What is SQL?

SQL (Structured Query Language) is a domain-specific programming language used to insert, update, query, and delete data in a database. It's also used to create and modify database schemas and manage database access and administration.

What is Structured Data?

Structured data is organized in a consistent, predefined format, typically alphanumeric. Examples include financial transactions and customer lists, usually stored in relational (SQL) databases.

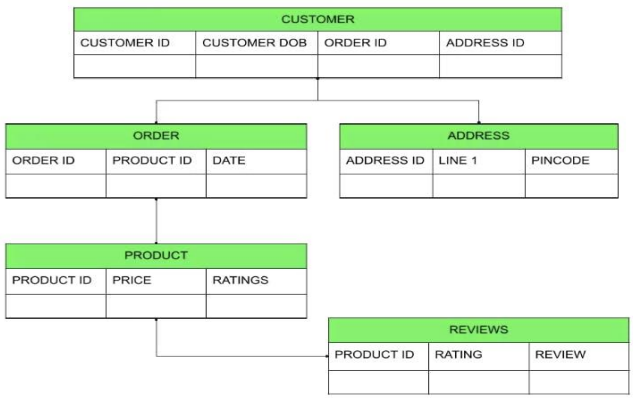
What is a SQL Database?

A SQL database uses SQL as its main language for data management. SQL APIs allow developers to execute and manage operations efficiently. These databases are categorized as **relational databases**, which store data in tables with rows and columns and support relationships through **foreign keys**.

Example:

An e-commerce database managing customer, product, and order information using related tables.

Relational databases use a **fixed schema**, meaning data must align with predefined formatting. They are ideal for structured or semi-structured data but not for unstructured formats.



Data is stored as tables with rows and columns in a relational database

Examples of SQL Databases:

- **Oracle**: Enterprise-grade RDBMS from Oracle Corporation.
- **MySQL**: Open-source RDBMS, known for ease of use and reliability.
- **PostgreSQL**: Feature-rich, open-source RDBMS supporting complex workloads.

- **MSSQL**: Microsoft's RDBMS used in enterprise applications.
- **SQLite**: Serverless, embedded RDBMS library with zero configuration.

NOT ONLY SQL (NOSQL)

What is NoSQL?

NoSQL databases are designed to store and manage **unstructured or semi-structured data**, allowing flexible schemas and formats like JSON, images, or audio.

What is Unstructured Data?

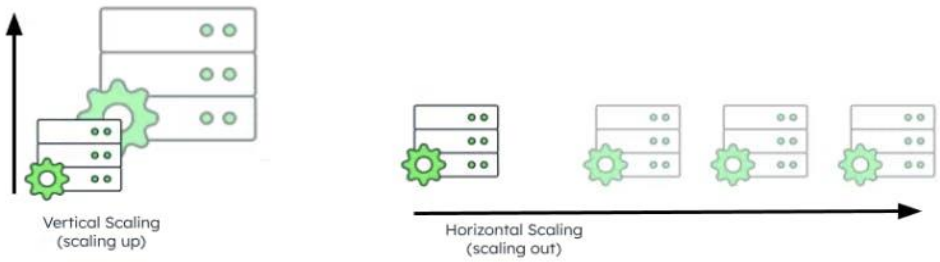
Data without a consistent structure—such as videos, images, or social media posts. It is often dynamic and doesn't conform to traditional database models.

What is a NoSQL Database?

NoSQL databases use **flexible schemas** and store data in **non-tabular** formats. They can ingest varied data types in their native formats.

Types of NoSQL Databases:

- **Document Databases** (e.g., MongoDB): Store data as documents (e.g., JSON-like).
- **Key-Value Stores** (e.g., ScyllaDB, AWS): Store data as unique key-value pairs.
- **Column-Family Stores** (e.g., Cassandra, HBase): Organize data in columns, ideal for wide datasets.
- **Graph Databases** (e.g., Neo4j, AWS): Store data in nodes and edges to represent relationships.



USE CASES OF SQL VS NOSQL DATABASE TECHNOLOGY

SQL databases are ideal for structured, relational data, ensuring transactional integrity and supporting complex queries. In contrast, NoSQL databases handle unstructured or semi-structured data with greater scalability and flexibility.

SQL Database Use Cases

SQL databases are well-suited for scenarios requiring ACID compliance and structured data management:

- **Regulatory Compliance**
  Their ACID-compliant nature makes them suitable for storing data subject to government or industry standards.
- **Transactional Databases**
  Used in systems where interactions between entities must be reliably recorded.
  Examples:
    - Retail PoS systems
    - Healthcare order databases
    - Banking and financial records
- **Enterprise Resource Planning (ERP)**
  Manage key business processes and operations.
  Examples:
    - HR databases
    - Supply chain systems
    - Risk management platforms

NoSQL Database Use Cases

Though traditionally considered non-ACID, some NoSQL databases like MongoDB do offer ACID compliance. NoSQL is preferred for managing unstructured data and high scalability.

- **Transactional Databases**
  Useful for unstructured data generated by transactions.
  Examples:
    - Patient records with scans/videos
    - Insurance files (e.g., accident photos)
    - Legal document repositories
- **Document Databases & Digital Asset Management (DAM)**
  Manage documents, images, and multimedia content.
  Examples:
    - Online legal libraries
    - Digital publishing platforms (e.g., Kindle)
    - Media streaming (e.g., Netflix)
    - Photo-sharing services (e.g., Instagram)
- **Graph and Network Analysis**
  Ideal for identifying complex relationships in connected data.
  Examples:
    - Social media metrics
    - Fraud detection
    - Knowledge graphs

Feature	SQL (Relational)	NoSQL (Non-relational)
Data Model	Tabular (rows and columns)	Document, key-value, column, or graph
Schema	Fixed, predefined	Flexible, dynamic
Data Types	Structured, semi-structured	Unstructured, semi-structured
Scalability	Vertical (adding power to one server)	Horizontal (adding more servers/nodes)
ACID Compliance	Strong support	Varies; some support ACID (e.g., MongoDB)
Best For	Transactions, financial, structured data	Big data, real-time apps, dynamic content

MongoDB. **FEATURES OF MONGODB (**NoSQL database):

1. **Document-Oriented Storage**
   - Stores data as **BSON documents** (Binary JSON), which allows embedded fields and arrays.
   - Example:

     ```
     {"name": "Alice", "age": 25, "skills": ["Python", "SQL"]}
     ```

2. **Schema Flexibility**
   - No fixed schema. Each document in a collection can have a **different structure**, making it ideal for evolving data models.
3. **High Performance**
   - Optimized for **read and write** operations with support for indexing, aggregation, and in-memory computing.
4. **Horizontal Scalability**
   - Uses **sharding** to distribute data across multiple machines, enabling large-scale horizontal scaling.
5. **Indexing Support**
   - Supports **primary, secondary**, **compound**, **geospatial**, **text**, and **hashed** indexes for fast queries.
6. **Aggregation Framework**
   - Allows powerful **data processing and transformation** using a pipeline concept (similar to SQL GROUP BY and JOIN operations).
7. **Replication for High Availability**
   - Supports **replica sets** (a group of servers with one primary and multiple secondaries) for **failover and redundancy**.
8. **Built-in Sharding**
   - Enables **automatic data partitioning** across shards, balancing the load and improving performance.
9. **ACID Transactions (since v4.0)**
   - Supports **multi-document ACID transactions**, especially important for financial or business-critical applications.
10. **Scalability and Big Data Handling**
    - Well-suited for **big data** and real-time analytics. Can handle **large volumes** of unstructured or semi-structured data.