Normalization Techniques

What is Normalization? Normalization is a database design technique used to minimize data redundancy and improve data integrity. It involves organizing data into multiple related tables to eliminate duplicate data and ensure consistency.

Why is Normalization Important?

* Reduces redundancy and avoids duplicate data.
* Improves data integrity and consistency.
* Simplifies data maintenance and updates.
* Enhances query performance in relational databases.

Normalization Forms:

1. **First Normal Form (1NF):** Ensures that all columns contain atomic (indivisible) values and each row has a unique identifier (primary key).
2. **Second Normal Form (2NF):** Achieved by removing partial dependencies, ensuring that all non-key attributes are fully dependent on the primary key.
3. **Third Normal Form (3NF):** Eliminates transitive dependencies by ensuring that non-key attributes depend only on the primary key.
4. **Boyce-Codd Normal Form (BCNF):** A stricter version of 3NF where every determinant is a candidate key.
5. **Fourth Normal Form (4NF):** Eliminates multi-valued dependencies, ensuring that a table does not contain multiple independent relationships.
6. **Fifth Normal Form (5NF):** Ensures that a table is split into smaller tables without losing any data, preventing join dependencies.
7. **Sixth Normal Form (6NF):** Rarely used, focuses on handling temporal databases by decomposing tables even further.

By applying these normalization techniques, databases achieve efficiency, consistency, and better scalability.

SQL Joins

What are SQL Joins? SQL Joins are used to retrieve data from multiple tables based on related columns. They help in combining rows from two or more tables in a meaningful way.

Types of Joins:

1. **INNER JOIN**: Retrieves records that have matching values in both tables. If there is no match, the row is not included in the result.
   * Syntax:

SELECT columns FROM table1

INNER JOIN table2 ON table1.column = table2.column;

1. **LEFT JOIN (LEFT OUTER JOIN)**: Retrieves all records from the left table and only the matching records from the right table. If there is no match, NULL values are returned for columns from the right table.
   * Syntax:

SELECT columns FROM table1

LEFT JOIN table2 ON table1.column = table2.column;

1. **RIGHT JOIN (RIGHT OUTER JOIN)**: Retrieves all records from the right table and only the matching records from the left table. If there is no match, NULL values are returned for columns from the left table.
   * Syntax:

SELECT columns FROM table1

RIGHT JOIN table2 ON table1.column = table2.column;

By using these join types, data from multiple tables can be efficiently queried and analyzed based on relationships.

## ✅ Basic MongoDB Interview Questions

### 1. ****What is MongoDB?****

MongoDB is a **NoSQL, document-oriented database** that stores data in **JSON-like BSON (Binary JSON)** format. It's schema-less and great for scalable, flexible applications.

### 2. ****What is a Document in MongoDB?****

A document is a **single record** in MongoDB, represented in **key-value** pairs, similar to a JavaScript object:

js

CopyEdit

{

name: "Vishnu",

age: 25,

skills: ["React", "Node"]

}

### 3. ****What is a Collection?****

A collection is a **group of documents**, like a table in SQL. Documents in a collection can have different fields (schema-less).

### 4. ****What are the Advantages of MongoDB?****

* Schema-less structure
* JSON-style documents
* Easy to scale (horizontal scaling)
* Fast read/write performance
* Good for real-time apps

### 5. ****Difference between SQL and MongoDB****

| SQL (RDBMS) | MongoDB (NoSQL) |
| --- | --- |
| Tables | Collections |
| Rows | Documents |
| Columns | Fields |
| Joins | Embedded documents/$lookup |
| Schema required | Flexible schema |

### 6. ****How to Insert Data into MongoDB?****

js

CopyEdit

db.users.insertOne({ name: "Vishnu", age: 25 });

db.users.insertMany([{ name: "Yash" }, { name: "Meghana" }]);

### 7. ****How to Query Documents?****

js

CopyEdit

db.users.find({ name: "Vishnu" }); // Find all with name Vishnu

db.users.findOne({ age: { $gt: 20 } }); // Find one with age > 20

### 8. ****How to Update a Document?****

js

CopyEdit

db.users.updateOne(

{ name: "Vishnu" },

{ $set: { age: 26 } }

);

### 9. ****How to Delete a Document?****

js

CopyEdit

db.users.deleteOne({ name: "Yash" });

db.users.deleteMany({ age: { $lt: 18 } });

### 10. ****What is an Index?****

An index improves query performance. Without it, MongoDB performs a full scan. You can create an index like:

js

CopyEdit

db.users.createIndex({ name: 1 }); // Ascending

### 11. ****What is ObjectId in MongoDB?****

\_id field is auto-generated and unique, using ObjectId, which is a 12-byte value (timestamp + machine id + process id + counter).

### 12. ****How do Relationships work in MongoDB?****

* **Embedded Documents** (like nesting)
* **Referencing** (using \_id and populate() in Mongoose)

### 13. ****What is Aggregation in MongoDB?****

Used for advanced data processing—like GROUP BY in SQL.

js

CopyEdit

db.users.aggregate([

{ $match: { age: { $gt: 20 } } },

{ $group: { \_id: "$age", count: { $sum: 1 } } }

]);

### 14. ****What is Mongoose?****

An ODM (Object Data Modeling) library for MongoDB in Node.js. It adds schema + model functionality.

### 15. ****How to Connect MongoDB to Node.js?****

js

CopyEdit

const mongoose = require("mongoose");

mongoose.connect("mongodb://localhost:27017/mydb", {

useNewUrlParser: true,

useUnifiedTopology: true

});

## ⚡ Want Some Tricky or Mid-Level MongoDB Questions?

Let me know, and I’ll give you:

* Real-world query questions
* Aggregation challenges
* Index tuning scenarios
* Mongoose validations and hooks

Let’s go 🔥 when you're ready!