Node.js with RDBMS

■ What is RDBMS?

- RDBMS (Relational Database Management System) stores data in tables (rows & columns).
- Examples: MySQL, PostgreSQL, Oracle, SQL Server.
- We use SQL (Structured Query Language) to query/update/manage data.

■ Why use RDBMS with Node.js?

- Node.js is great for building backend APIs.
- RDBMS provides a reliable, structured, transactional data store.
- Together, they form the backbone of most web apps (e.g., banking apps, e-commerce, ERP).

■ How to Connect Node.js with RDBMS?

1. Install dependencies

```
npm init -y
npm install express mysql2
```

```
npm install pg
```

2. Example: Node.js + MySQL

```
const express = require("express");
const mysql = require("mysql2");

const app = express();
app.use(express.json());

// Create DB connection
const db = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "password",
  database: "testdb"
});

// Connect
db.connect(err => {
  if (err) throw err;
  console.log("MySQL Connected...");
});

// API route - Get all users
app.get("/users", (req, res) => {
  db.query("SELECT * FROM users", (err, results) => {
    if (err) throw err;
    res.json(results);
  });
});
```

```
// API route - Insert user
app.post("/users", (req, res) => {
  const { name, email } = req.body;
  db.query("INSERT INTO users (name, email) VALUES (?, ?)", [name, email], (err, result) => {
    if (err) throw err;
    res.json({ id: result.insertId, name, email });
  });
});

app.listen(3000, () => console.log("Server running on port 3000"));
```

3. Example: Node.js + PostgreSQL

```
const express = require("express");
const { Pool } = require("pg");

const app = express();
app.use(express.json());

const pool = new Pool({
  user: "postgres",
  host: "localhost",
  database: "testdb",
  password: "password",
  port: 5432,
});

// Get all users
app.get("/users", async (req, res) => {
  const result = await pool.query("SELECT * FROM users");
  res.json(result.rows);
});

// Insert user
app.post("/users", async (req, res) => {
  const { name, email } = req.body;
  const result = await pool.query(
    "INSERT INTO users (name, email) VALUES ($1, $2) RETURNING *",
    [name, email]
  );
  res.json(result.rows[0]);
});

app.listen(3000, () => console.log("Server running on port 3000"));
```

■ When to Use Node.js with RDBMS?

■ Banking systems (transactions, ACID properties)
■ E-commerce (products, orders, payments)
■ Employee/Payroll systems
■ Any application requiring structured relational data.