Database- Day -2: MySQL

Normalization, select queries, joins:

Normalization:

Normalization in databases is a process used to organize a database into tables and columns. The idea is to **reduce redundancy and improve data integrity.**

Table: StudentCourses

```
+------------+-----------+------------+
| StudentID  | StudentName   | Course   |
+------------+-----------+------------+
| 1            | John Doe       | Math     |
| 1            | John Doe       | Science  |
| 2            | Jane Smith     | Math     |
| 3            | Jim Brown      | History  |
+------------+-----------+------------+
```

We need to **eliminate repeating groups.** Each field must contain only atomic (indivisible) values, and **each record needs to be unique**.

Students

```
+------------+-----------+
| StudentID  | StudentName |
+------------+-----------+
| 1        | John Doe    |
| 2        | Jane Smith  |
| 3        | Jim Brown   |
+------------+-----------+
```

Courses

```
+------------+-----------+
| CourseID   | CourseName |
+------------+-----------+
| 1          | Math      |
| 2          | Science   |
| 3          | History   |
+------------+-----------+
```

StudentCourses

```
+------------+-----------+
| StudentID  | CourseID  |
+------------+-----------+
| 1          | 1         |
| 1          | 2         |
| 2          | 1         |
| 3          | 3         |
+------------+-----------+
```

select queries:

some examples of SELECT queries in MySQL:

1. mysql> select * from employees;
2. mysql> select first_name, email from employees; **(specific columns)**
3. mysql> select first_name, email from employees where emp_id=104;
4. mysql> select first_name, email from employees order by first_name;
5. mysql> select first_name, email from employees order by first_name limit 2;
6. mysql> select designation, count(*) from employees group by designation;
7. mysql> select distinct designation from employees;
8. mysql> select * from employees where first_name like 's%';
9. mysql> select * from employees where first_name like 'sa%';
10. mysql> select * from employees where first_name like '%k';

Joins:

A JOIN in MySQL is used to **combine rows from two or more tables**, based on a **related column** between them.

Types:

1. **INNER JOIN** - Retrieve records from both tables where there is a match
2. LEFT JOIN (or LEFT OUTER JOIN) - Retrieve all records from the left table and matching records from the right table
3. RIGHT JOIN (or RIGHT OUTER JOIN) - Retrieve all records from the right table and matching records from the left table
4. FULL JOIN (or FULL OUTER JOIN) - Retrieve all records when there is a match in either the left or right table
5. CROSS JOIN - Create a **Cartesian** product of both tables (every combination of rows from both tables)
6. SELF JOIN - Join a table with itself (not used in real time)

Ex:

```
CREATE TABLE departments (id INT PRIMARY KEY,name VARCHAR(255));

INSERT INTO departments (id, name) VALUES
(1, 'HR'),(2, 'IT'),(3, 'Finance');

select * from departments;

CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   first_name VARCHAR(255) NOT NULL,
   last_name VARCHAR(255) NOT NULL,
   department_id INT,
   FOREIGN KEY (department_id) REFERENCES departments(id)
);

INSERT INTO employees (employee_id, first_name, last_name,
department_id) VALUES
(1, 'suresh', 'vikram', 1),
(2, 'rithik', 'suresh', 2),
(3, 'sathvik', 'suresh', 3),
(4, 'renu', 'krishnan', 2),
(5, 'mini', 'kumar', 1);
```

```
select * from employees;

EG:

-- InnerJoin
SELECT employees.first_name, employees.last_name, departments.name
FROM employees
INNER JOIN departments ON employees.department_id = departments.id;

-- Left join
SELECT employees.first_name, employees.last_name, departments.name
FROM employees
LEFT JOIN departments ON employees.department_id = departments.id;

-- Right join
SELECT employees.first_name, employees.last_name, departments.name
FROM employees
RIGHT JOIN departments ON employees.department_id = departments.id;

-- Full Join
SELECT employees.first_name, employees.last_name, departments.name
FROM employees
JOIN departments ON employees.department_id = departments.id;

-- cross join
SELECT employees.first_name, employees.last_name, departments.name
FROM employees CROSS JOIN departments;

describe employees;
drop table employees;
drop table departments;
```

DB model design: (reverse engg) example related to task

Eg: (show ER diagram in workbench - reverse engineer)

```
CREATE TABLE departments (id INT PRIMARY KEY,name VARCHAR(255));
CREATE TABLE employees (
        employee_id INT PRIMARY KEY,
        first_name VARCHAR(255) NOT NULL,
        last_name VARCHAR(255) NOT NULL,
        department_id INT,
        FOREIGN KEY (department_id) REFERENCES departments(id)
      );
```

```
create table salary (id int primary key, department_id int, employee_id
int,
foreign key (department_id) references departments(id),
foreign key (employee_id) references employees(employee_id));
```

Intro to MongoDB & installation of Mongodb:

Introduction to MongoDB : https://www.mongodb.com/basics

MongoDB is a popular, open-source **NoSQL database management system** known for its flexibility, scalability, and ease of use. It is designed to handle large volumes of **unstructured or semi-structured data**, making it well-suited for a wide range of applications

Key Concepts:

- Document- Oriented - it stores data in **JSON-like documents**
- Collections- data into collections (**tables**)
- NoSQL- (Not Only SQL) database ,No fixed schemas/structure
- Scalability- handling large datasets.(load balance)

Main Feature:

- Query Language - MongoDB provides a powerful query language
- Indexes - improve query performance.
- Aggregation – **pipeline** ,performing complex data transformations
- Replication - transaction
- Sharding/clustering - multiple servers or nodes

Relational & Non-Relational Databases Difference :

Relational and non-relational databases are two different categories of database management systems

- **Relational Databases (RDBMS):** In relational databases, data is organized into **structured tables with predefined schemas**. **Tables** consist of rows (records) and columns (fields), and data must conform to the schema.
- **Non-Relational Databases (NoSQL):** Non-relational databases are more flexible in terms of data modeling. They can store data in various formats, including documents**, key-value pairs**, wide-column stores, and graph databases. There is no strict schema, allowing for more dynamic and **unstructured data**.

Examples:

Relational Databases (RDBMS):

MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.

Non-Relational Databases (NoSQL):

MongoDB, Cassandra, Amazon DynamoDB, and Neo4j.

When to Use MongoDB:

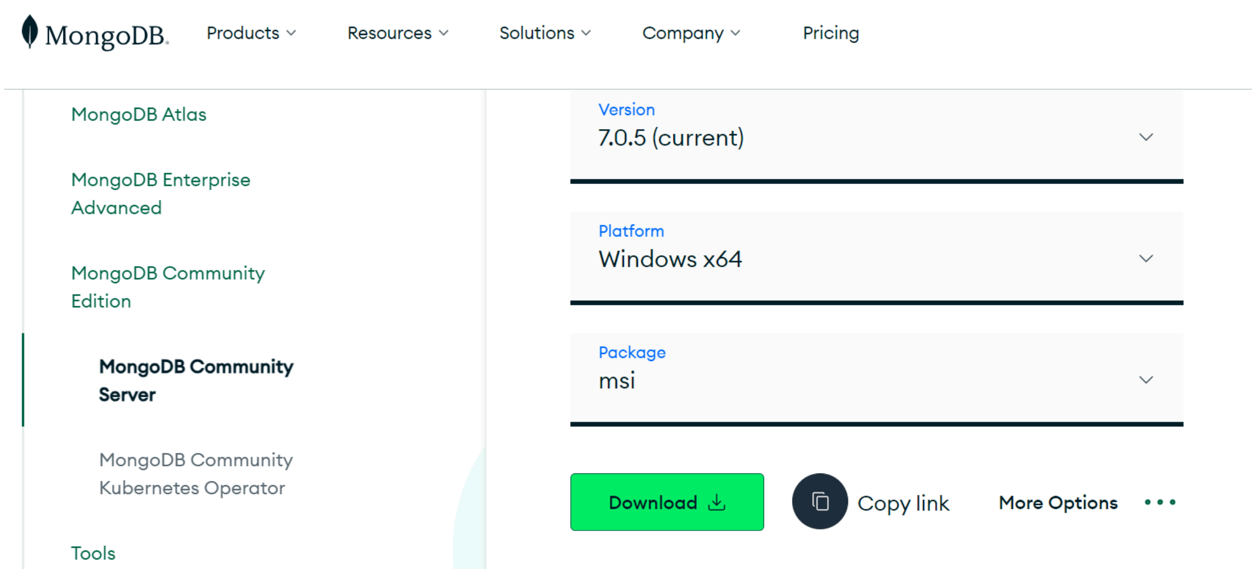
MongoDB works best with **unstructured data,** so it's great for Big Data systems.

if you're using cloud computing. MongoDB is ideal for cloud computing.

MongoDB Installation:

Video ref: https://youtu.be/PHXhuc8MwRw?si=96wcqGGRewKXk8wV

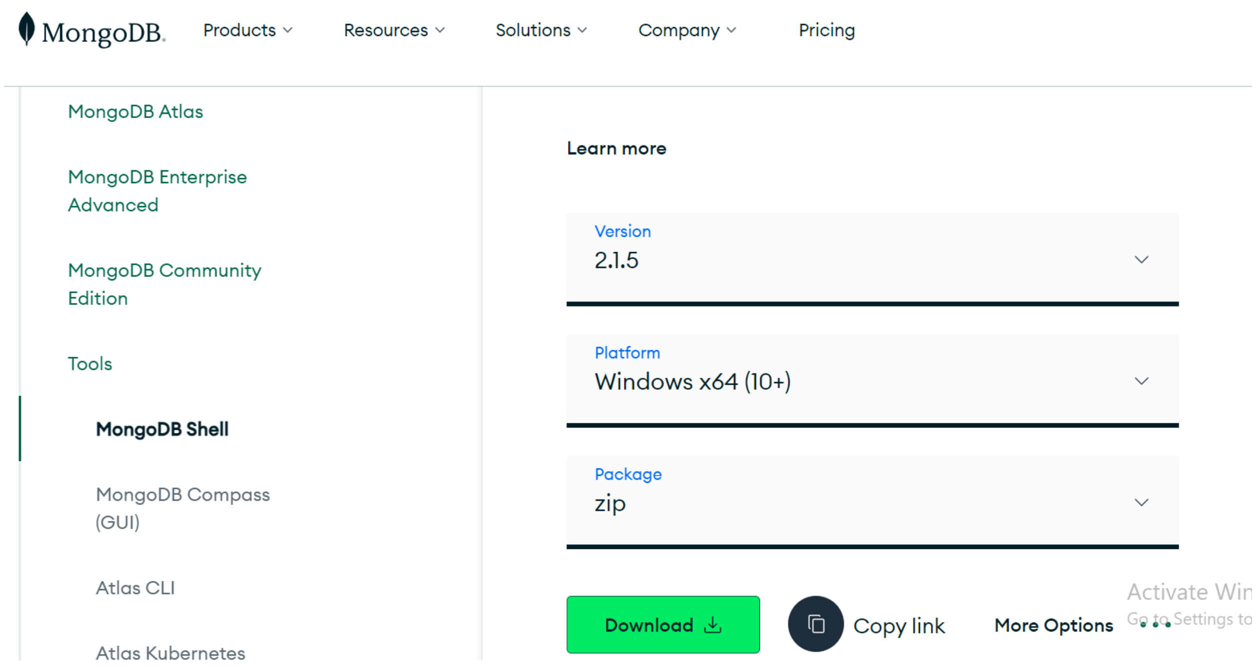
https://www.mongodb.com/try/download/community



Version
7.0.5 (current)

Platform
Windows x64

Package
msi



Version: 2.1.5

Platform: Windows x64 (10+)

Package: zip

install mongoDb7.5th version or any version & Compass

tools MongoDbShell

setting env path both server and client

Command prompt:

```
        C:\Users\Digital Suppliers>mongod --version
db version v7.0.1
Build Info: {
    "version": "7.0.1",
    "gitVersion": "425a0454d12f2664f9e31002bbe4a386a25345b5",
    "modules": [],
    "allocator": "tcmalloc",

    "environment": {

        "distmod": "windows",

        "distarch": "x86_64",

        "target_arch": "x86_64"

        }

}
```

Service check - > service-> mongoDB is running or not.

Open another command prompt:

step:1

```
        C:\Users\Digital Suppliers>mongosh

        Current Mongosh Log ID: 6509f4c35fa3ec75c1fcffbe

        Connecting to:
        mongodb://127.0.0.1:27017/?directConnection=true&serverSele
        ctionTimeoutMS=2000&appName=mongosh+1.10.6

        Using MongoDB:      7.0.1

        Using Mongosh:      1.10.6

        For mongosh info see:
https://docs.mongodb.com/mongodb-shell/
```

```
To help improve our products, anonymous usage data is
collected and sent to MongoDB periodically
(https://www.mongodb.com/legal/privacy-policy).

You can opt-out by running the disableTelemetry() command.

The server generated these startup warnings when booting

   2023-09-20T00:36:31.118+05:30: Access control is not
enabled for the database. Read and write access to data and
configuration is unrestricted

test>
```

step:2

```
test> show dbs;

admin   40.00 KiB

config  60.00 KiB

local   40.00 KiB

test>
```