SQL_DB_and_Tables_creation

May 25, 2024

0.0.1 Create Database and Tables inside

To create an employee table that can support all the SQL queries mentioned, we'll need a table with a variety of columns to cover different scenarios, including employee details, departments, salaries, hire dates, and other necessary fields.

Here's the SQL code to create such a table and populate it with sample data:

```sql
-- Create the departments table
CREATE TABLE departments (
    department_id INT PRIMARY KEY AUTO_INCREMENT,
    department_name VARCHAR(255) NOT NULL
);

-- Create the employees table
CREATE TABLE employees (
    employee_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    phone_number VARCHAR(20),
    hire_date DATE NOT NULL,
    job_title VARCHAR(255) NOT NULL,
    salary DECIMAL(10, 2) NOT NULL CHECK (salary > 0),
    department_id INT,
    manager_id INT,
    birth_date DATE,
    address VARCHAR(255),
    city VARCHAR(255),
    state VARCHAR(255),
    country VARCHAR(255),
    postal_code VARCHAR(20),
    start_date DATE,
    end_date DATE,
    json_data JSON,  -- For JSON data example (use TEXT if JSON is not supported)
    xml_data TEXT,   -- For XML data example
    FOREIGN KEY (department_id) REFERENCES departments(department_id),
    FOREIGN KEY (manager_id) REFERENCES employees(employee_id)
);
```

```sql
-- Insert sample data into departments table
INSERT INTO departments (department_name) VALUES
('HR'),
('Finance'),
('IT'),
('Marketing'),
('Sales');

-- Insert sample data into employees table
INSERT INTO employees (first_name, last_name, email, phone_number, hire_date, job_title, salary,
('Alice', 'Johnson', 'alice.johnson@example.com', '555-1234', '2020-03-15', 'HR Manager', 75000,
('Bob', 'Smith', 'bob.smith@example.com', '555-5678', '2019-07-01', 'Financial Analyst', 60000,
('Carol', 'White', 'carol.white@example.com', '555-8765', '2021-01-20', 'IT Specialist', 80000,
('David', 'Brown', 'david.brown@example.com', '555-4321', '2018-05-10', 'Marketing Manager', 850
('Eve', 'Davis', 'eve.davis@example.com', '555-8765', '2022-02-28', 'Sales Associate', 55000, 5,

-- Creating index on email column
CREATE INDEX idx_email ON employees(email);

-- Create some sample projects data for JOIN examples
CREATE TABLE projects (
    project_id INT PRIMARY KEY AUTO_INCREMENT,
    project_name VARCHAR(255) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE,
    employee_id INT,
    FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
);

-- Insert sample data into projects table
INSERT INTO projects (project_name, start_date, end_date, employee_id) VALUES
('Project Alpha', '2021-01-01', '2021-06-30', 1),
('Project Beta', '2020-05-01', '2020-12-31', 2),
('Project Gamma', '2021-07-01', NULL, 3),
('Project Delta', '2019-01-01', '2019-12-31', 4),
('Project Epsilon', '2022-01-01', NULL, 5);
```

0.0.2 Notes:

- **AUTO_INCREMENT** is used for automatically incrementing primary keys in MySQL.
- **VARCHAR** is used for variable-length strings.
- **JSON** is used for storing JSON data, but if your MySQL version does not support it, you can use `TEXT` instead.
- **Indexes** and **foreign keys** are added for better performance and data integrity.

This should create the necessary tables and allow you to run a variety of SQL queries on them.

0.0.3 Explanation

- **Tables**: The `employees` and `departments` tables are created with various columns to handle different SQL scenarios.
- **Sample Data**: Sample data is inserted into the `departments` and `employees` tables to provide data for running queries.
- **Indexes**: An index is created on the `email` column to support query optimization.
- **Foreign Keys**: Foreign key constraints ensure referential integrity between `employees` and `departments`, and between `employees` themselves for the manager relationship.
- **Additional Tables**: A `projects` table is included to provide data for JOIN examples.

[ ]: