**Project Title: MySQL Employee Database: Tracking Sales and Relationships**

**Introduction**

The MySQL Employee Database project is a comprehensive initiative designed to enhance organizational efficiency through the creation of a robust system for managing employee information, tracking sales, and handling relationships within the corporate structure. In today's dynamic business environment, effective employee management and insightful sales tracking are critical components for the success of any organization. This project addresses these needs by leveraging the power of MySQL, a widely used relational database management system.

**Background:**

The necessity for an organized and efficient employee data management system stem from the complexities of modern workplaces. As organizations grow, the volume of employee information increases, making it crucial to have a centralized database that ensures accurate and easily accessible data. Additionally, the project recognizes the significance of tracking sales and managing client relationships, essential aspects for businesses to thrive in competitive markets.

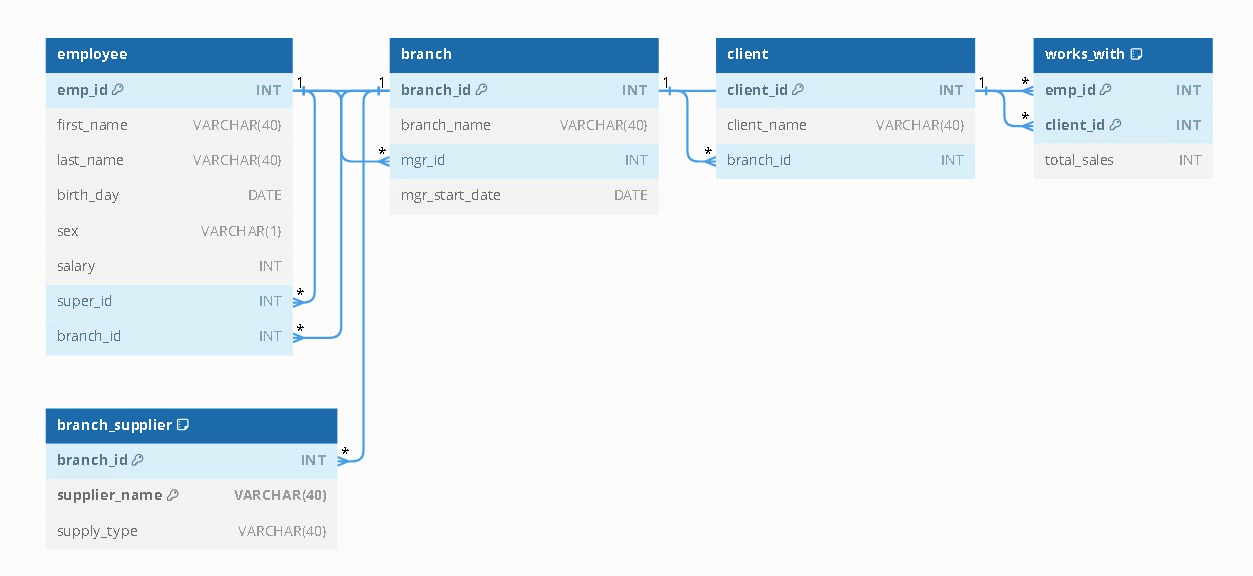
**Objectives:**

The primary objectives of the MySQL Employee Database project are twofold: to streamline the management of employee data and to facilitate the tracking of sales and client relationships. By achieving these objectives, the project aims to provide organizations with a reliable and scalable solution for optimizing their internal processes.

1. **Efficiently Manage Employee Data:** The project seeks to simplify the process of managing employee information. This includes storing and retrieving employee details, such as names, birthdates, and branch assignments, in a structured and organized manner. By centralizing this information, the system aims to eliminate redundancy and improve the overall efficiency of HR processes.
2. **Track Sales and Client Relationships:** An integral aspect of the project is the ability to track sales and manage client relationships. Through the Works\_With table, the system connects employees with clients, recording total sales for each interaction. This functionality provides valuable insights into the organization's revenue streams and client engagement, contributing to informed decision-making.
3. **Analyze Branch Performance:** The project incorporates features for analyzing branch performance based on sales data. By aggregating and summarizing sales information at the branch level, organizations can gain a comprehensive understanding of their overall performance and identify areas for improvement.

**Entity-Relationship Diagram (ERD)**

The Entity-Relationship Diagram (ERD) serves as the foundational blueprint for the MySQL Employee Database project, illustrating the relationships between key entities and their attributes. The ERD showcases the interconnectedness of entities, facilitating a visual understanding of the database's structure.



The main entities depicted in the ERD include:

* **Employee:** Represents individual employees with attributes such as emp\_id, first\_name, last\_name, birth\_day, sex, salary, super\_id, and branch\_id.
* **Branch:** Captures information about different branches within the organization, with attributes like branch\_id, branch\_name, mgr\_id, and mgr\_start\_date.
* **Client:** Encompasses client details with attributes client\_id, client\_name, and branch\_id.
* **Works\_With:** Establishes the relationship between employees and clients, tracking total\_sales as a measure of their interaction.
* **Branch\_Supplier:** Represents suppliers associated with specific branches, with attributes branch\_id, supplier\_name, and supply\_type.

**Database Schema**

**Employee Table**

The Employee table serves as a core component, storing comprehensive information about each employee. Attributes include emp\_id as the primary key, along with first\_name, last\_name, birth\_day, sex, salary, super\_id (indicating the supervisor), and branch\_id (denoting the employee's branch assignment).

**Branch Table**

The Branch table encapsulates data related to organizational branches. It includes attributes such as branch\_id (primary key), branch\_name, mgr\_id (identifying the branch manager), and mgr\_start\_date.

**Client Table**

The Client table manages client-specific details, including client\_id as the primary key, client\_name, and branch\_id (indicating the associated branch).

**Works\_With Table**

The Works\_With table establishes the relationship between employees and clients, facilitating the tracking of total sales. It includes emp\_id and client\_id as composite primary keys, along with the total\_sales attribute.

**Branch\_Supplier Table**

The Branch\_Supplier table manages supplier information associated with specific branches. It consists of branch\_id and supplier\_name as composite primary keys, along with the supply\_type attribute.

**Implementation**

**Database Setup**

The project is exclusively implemented using MySQL, a powerful relational database management system. The database setup involves creating tables to store employee, branch, client, works\_with, and branch\_supplier information. Primary keys and foreign keys are defined to establish relationships between tables and ensure data integrity.

**Code:**   
CREATE TABLE employee (

emp\_id INT PRIMARY KEY,

first\_name VARCHAR(40),

last\_name VARCHAR(40),

birth\_day DATE,

sex VARCHAR(1),

salary INT,

super\_id INT,

branch\_id INT

);

CREATE TABLE branch (

branch\_id INT PRIMARY KEY,

branch\_name VARCHAR(40),

mgr\_id INT,

mgr\_start\_date DATE,

FOREIGN KEY(mgr\_id) REFERENCES employee(emp\_id) ON DELETE SET NULL

);

ALTER TABLE employee

ADD FOREIGN KEY(branch\_id)

REFERENCES branch(branch\_id)

ON DELETE SET NULL;

ALTER TABLE employee

ADD FOREIGN KEY(super\_id)

REFERENCES employee(emp\_id)

ON DELETE SET NULL;

CREATE TABLE client (

client\_id INT PRIMARY KEY,

client\_name VARCHAR(40),

branch\_id INT,

FOREIGN KEY(branch\_id) REFERENCES branch(branch\_id) ON DELETE SET NULL

);

CREATE TABLE works\_with (

emp\_id INT,

client\_id INT,

total\_sales INT,

PRIMARY KEY(emp\_id, client\_id), FOREIGN KEY(emp\_id) REFERENCES employee(emp\_id) ON DELETE CASCADE,

FOREIGN KEY(client\_id) REFERENCES client(client\_id) ON DELETE CASCADE

);

CREATE TABLE branch\_supplier (

branch\_id INT,

supplier\_name VARCHAR(40),

supply\_type VARCHAR(40),

PRIMARY KEY(branch\_id, supplier\_name),

FOREIGN KEY(branch\_id) REFERENCES branch(branch\_id) ON DELETE CASCADE

);

-- -----------------------------------------------------------------------------

-- Corporate

INSERT INTO employee VALUES(100, 'David', 'Wallace', '1967-11-17', 'M', 250000, NULL, NULL);

INSERT INTO branch VALUES(1, 'Corporate', 100, '2006-02-09');

UPDATE employee

SET branch\_id = 1

WHERE emp\_id = 100;

INSERT INTO employee VALUES(101, 'Jan', 'Levinson', '1961-05-11', 'F', 110000, 100, 1);

-- Scranton

INSERT INTO employee VALUES(102, 'Michael', 'Scott', '1964-03-15', 'M', 75000, 100, NULL);

INSERT INTO branch VALUES(2, 'Scranton', 102, '1992-04-06');

UPDATE employee

SET branch\_id = 2

WHERE emp\_id = 102;

INSERT INTO employee VALUES(103, 'Angela', 'Martin', '1971-06-25', 'F', 63000, 102, 2);

INSERT INTO employee VALUES(104, 'Kelly', 'Kapoor', '1980-02-05', 'F', 55000, 102, 2);

INSERT INTO employee VALUES(105, 'Stanley', 'Hudson', '1958-02-19', 'M', 69000, 102, 2);

-- Stamford

INSERT INTO employee VALUES(106, 'Josh', 'Porter', '1969-09-05', 'M', 78000, 100, NULL);

INSERT INTO branch VALUES(3, 'Stamford', 106, '1998-02-13');

UPDATE employee

SET branch\_id = 3

WHERE emp\_id = 106;

INSERT INTO employee VALUES(107, 'Andy', 'Bernard', '1973-07-22', 'M', 65000, 106, 3);

INSERT INTO employee VALUES(108, 'Jim', 'Halpert', '1978-10-01', 'M', 71000, 106, 3);

-- BRANCH SUPPLIER

INSERT INTO branch\_supplier VALUES(2, 'Hammer Mill', 'Paper');

INSERT INTO branch\_supplier VALUES(2, 'Uni-ball', 'Writing Utensils');

INSERT INTO branch\_supplier VALUES(3, 'Patriot Paper', 'Paper');

INSERT INTO branch\_supplier VALUES(2, 'J.T. Forms & Labels', 'Custom Forms');

INSERT INTO branch\_supplier VALUES(3, 'Uni-ball', 'Writing Utensils');

INSERT INTO branch\_supplier VALUES(3, 'Hammer Mill', 'Paper');

INSERT INTO branch\_supplier VALUES(3, 'Stamford Lables', 'Custom Forms');

-- CLIENT

INSERT INTO client VALUES(400, 'Dunmore Highschool', 2);

INSERT INTO client VALUES(401, 'Lackawana Country', 2);

INSERT INTO client VALUES(402, 'FedEx', 3);

INSERT INTO client VALUES(403, 'John Daly Law, LLC', 3);

INSERT INTO client VALUES(404, 'Scranton Whitepages', 2);

INSERT INTO client VALUES(405, 'Times Newspaper', 3);

INSERT INTO client VALUES(406, 'FedEx', 2);

-- WORKS\_WITH

INSERT INTO works\_with VALUES(105, 400, 55000);

INSERT INTO works\_with VALUES(102, 401, 267000);

INSERT INTO works\_with VALUES(108, 402, 22500);

INSERT INTO works\_with VALUES(107, 403, 5000);

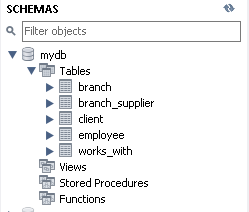
INSERT INTO works\_with VALUES(108, 403, 12000);

INSERT INTO works\_with VALUES(105, 404, 33000);

INSERT INTO works\_with VALUES(107, 405, 26000);

INSERT INTO works\_with VALUES(102, 406, 15000);

INSERT INTO works\_with VALUES(105, 406, 130000);



**Queries and Reports**

Various queries are implemented to retrieve employee information, calculate total sales, count employees in each branch, and establish client-employee relationships.

1. **Retrieve the names of all employees.**

SELECT first\_name, last\_name FROM employee;

1. **Get the list of all branches.**

SELECT branch\_name FROM branch;

1. **Display the names of clients.**

SELECT client\_name FROM client;

1. **List all employees who have a supervisor.**

SELECT first\_name, last\_name FROM employee WHERE super\_id IS NOT NULL;

1. **Find the total number of clients.**

SELECT COUNT(\*) AS total\_clients FROM client;

1. **Retrieve the branch name where employee with emp\_id 107 works.**

SELECT branch\_name FROM branch

WHERE branch\_id = (SELECT branch\_id FROM employee

WHERE emp\_id = 107);

1. **Get the names of employees who were born in 1970.**

SELECT first\_name, last\_name FROM employee

WHERE YEAR(birth\_day) = 1970;

1. **List all clients along with the branch names they are associated with.**

SELECT client.client\_name, branch.branch\_name FROM client

JOIN branch ON client.branch\_id = branch.branch\_id;

1. **Calculate the average salary of all employees.**

SELECT AVG(salary) AS average\_salary FROM employee;

1. **Find the employee with the highest salary.**

SELECT emp\_id, first\_name, last\_name, salary FROM employee ORDER BY salary DESC LIMIT 1;

1. **Count the number of employees in each branch.**

SELECT branch\_id, COUNT(\*) AS employee\_count FROM employee GROUP BY branch\_id;

1. **Retrieve the names of employees who work with the client "FedEx."**

SELECT first\_name, last\_name FROM employee

JOIN works\_with ON employee.emp\_id = works\_with.emp\_id

JOIN client ON works\_with.client\_id = client.client\_id

WHERE client.client\_name = 'FedEx';

1. **Display the branch names and the total number of employees in each branch.**

SELECT branch\_name, COUNT(\*) AS total\_employees FROM branch

JOIN employee ON branch.branch\_id = employee.branch\_id

GROUP BY branch\_name;

1. **Retrieve the names of employees who do not have a supervisor.**

SELECT first\_name, last\_name FROM employee

WHERE super\_id IS NULL;

1. **Identify the employee with the highest total sales.**

SELECT emp\_id, first\_name, last\_name, SUM(total\_sales) AS total\_sales FROM employee

LEFT JOIN works\_with ON employee.emp\_id = works\_with.emp\_id GROUP BY emp\_id, first\_name, last\_name

ORDER BY total\_sales DESC LIMIT 1;

1. **Find the branch that has the highest average salary among its employees.**

SELECT branch\_id, AVG(salary) AS avg\_salary FROM employee GROUP BY branch\_id

ORDER BY avg\_salary DESC LIMIT 1;

1. **Retrieve the names of employees who work with more than one client.**

SELECT first\_name, last\_name FROM employee

JOIN works\_with ON employee.emp\_id = works\_with.emp\_id GROUP BY employee.emp\_id

HAVING COUNT(works\_with.client\_id) > 1;

1. **List the names of employees who were born before their supervisors.**

SELECT e1.first\_name, e1.last\_name FROM employee e1

JOIN employee e2 ON e1.super\_id = e2.emp\_id

WHERE e1.birth\_day < e2.birth\_day;

1. **Calculate the total sales for each client.**

SELECT client\_id, client\_name, SUM(total\_sales) AS total\_client\_sales FROM works\_with

JOIN client ON works\_with.client\_id = client.client\_id

GROUP BY client\_id, client\_name;

1. **Identify the branch with the highest total sales.**

SELECT employee.emp\_id, employee.first\_name, employee.last\_name, SUM(works\_with.total\_sales) AS total\_sales FROM employee

LEFT JOIN works\_with ON employee.emp\_id = works\_with.emp\_id

GROUP BY employee.emp\_id, employee.first\_name, employee.last\_name

ORDER BY total\_sales DESC LIMIT 1;

GitHub Link: [KarthicV358/MySQL-Employee-Database-Tracking-Sales-and-Relationships (github.com)](https://github.com/KarthicV358/MySQL-Employee-Database-Tracking-Sales-and-Relationships)

**Conclusion**

In conclusion, the MySQL Employee Database project represents a successful endeavour in creating an efficient and comprehensive system for managing employee information, tracking sales, and analyzing organizational relationships. Leveraging the power of MySQL and SQL, the project has established a well-structured database with tables for Employee, Branch, Client, Works\_With, and Branch\_Supplier.

The database design, illustrated through the Entity-Relationship Diagram and detailed in the Database Schema section, forms the foundation for the project's success. It provides a clear representation of the interconnected entities and relationships, ensuring that the database captures and maintains data in a logical and organized manner.

The implementation phase, focused exclusively on MySQL and SQL, has resulted in a functional and responsive database system. The setup involved careful consideration of foreign key relationships, indexing key fields, and the strategic population of sample data. The execution of SQL queries, detailed in the Queries and Reports section, demonstrates the system's capability to retrieve employee information, calculate sales, and analyze branch performance.

By achieving its objectives of efficient employee data management, sales tracking, and relationship analysis, the MySQL Employee Database project offers a valuable tool for organizations seeking to optimize their internal processes. As technology evolves, this project serves as a testament to the enduring relevance and effectiveness of MySQL and SQL in developing robust database solutions. Future enhancements and optimizations can further refine and expand the project's capabilities, ensuring its continued utility in dynamic business environments.Top of Form