To manage employees, departments, payroll, and performance evaluations, you can design the following SQL schema with necessary tables, stored procedures, triggers, and queries.

**Step 1:** Database Schema

1. Departments Table:

This table stores information about the departments.

CREATE TABLE departments (

department\_id INT PRIMARY KEY AUTO\_INCREMENT,

department\_name VARCHAR(100) NOT NULL,

department\_budget DECIMAL(15, 2) NOT NULL

);

2. Employees Table:

This table stores information about employees.

CREATE TABLE employees (

employee\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

department\_id INT,

salary DECIMAL(15, 2),

hire\_date DATE,

performance\_rating INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

3. Payroll Table:

This table stores payroll information for employees.

CREATE TABLE payroll (

payroll\_id INT PRIMARY KEY AUTO\_INCREMENT,

employee\_id INT,

pay\_date DATE,

salary DECIMAL(15, 2),

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

4. Performance Evaluations Table:

This table stores employee performance reviews.

CREATE TABLE performance\_evaluations (

evaluation\_id INT PRIMARY KEY AUTO\_INCREMENT,

employee\_id INT,

evaluation\_date DATE,

score INT,

feedback TEXT,

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

**Step 2:** Stored Procedures

1. Updating Payroll:

This stored procedure will update the salary in the payroll and employees table when there is a change in the payroll.

DELIMITER $$

CREATE PROCEDURE update\_payroll(

IN emp\_id INT,

IN new\_salary DECIMAL(15, 2)

)

BEGIN

-- Update employee salary

UPDATE employees

SET salary = new\_salary

WHERE employee\_id = emp\_id;

-- Insert into payroll table

INSERT INTO payroll (employee\_id, pay\_date, salary)

VALUES (emp\_id, CURDATE(), new\_salary);

END $$

DELIMITER ;

2. Handling Performance Reviews:

This stored procedure inserts a performance review for an employee.

DELIMITER $$

CREATE PROCEDURE add\_performance\_review(

IN emp\_id INT,

IN score INT,

IN feedback TEXT

)

BEGIN

INSERT INTO performance\_evaluations (employee\_id, evaluation\_date, score, feedback)

VALUES (emp\_id, CURDATE(), score, feedback);

-- Update performance rating of the employee

UPDATE employees

SET performance\_rating = score

WHERE employee\_id = emp\_id;

END $$

DELIMITER ;

**Step 3:** Triggers

1. Update Department Budget when Payroll Changes:

This trigger ensures that the department's budget is adjusted whenever an employee's salary is updated.

DELIMITER $$

CREATE TRIGGER update\_department\_budget

AFTER UPDATE ON employees

FOR EACH ROW

BEGIN

DECLARE total\_payroll DECIMAL(15, 2);

-- Calculate total payroll for the department

SELECT SUM(salary) INTO total\_payroll

FROM employees

WHERE department\_id = OLD.department\_id;

-- Update department budget

UPDATE departments

SET department\_budget = total\_payroll

WHERE department\_id = OLD.department\_id;

END $$

DELIMITER ;

**Step 4:** SQL Queries

1. Generate Employee Performance Reports:

This query generates a report of employee performance reviews, including the employee details and their most recent performance review.

SELECT e.employee\_id, e.first\_name, e.last\_name, e.department\_id, e.salary, e.performance\_rating, p.evaluation\_date, p.score, p.feedback

FROM employees e

JOIN performance\_evaluations p ON e.employee\_id = p.employee\_id

WHERE p.evaluation\_date = (

SELECT MAX(evaluation\_date)

FROM performance\_evaluations

WHERE employee\_id = e.employee\_id

)

ORDER BY e.department\_id, e.last\_name;

2. Generate Payroll Summary:

This query generates a payroll summary for all employees.

SELECT e.employee\_id, e.first\_name, e.last\_name, e.department\_id, p.pay\_date, p.salary

FROM employees e

JOIN payroll p ON e.employee\_id = p.employee\_id

ORDER BY p.pay\_date DESC;

3. Generate Department Budget Report:

This query shows the total payroll of each department along with the current department budget.

SELECT d.department\_name, d.department\_budget, SUM(e.salary) AS total\_payroll

FROM departments d

JOIN employees e ON d.department\_id = e.department\_id

GROUP BY d.department\_id;

Example Workflow:

When an employee gets a salary increase, you call the stored procedure update\_payroll to update the salary and record the change in the payroll table.

If an employee undergoes a performance evaluation, you call the add\_performance\_review stored procedure to record the review and update the performance rating.

Whenever an employee's salary changes, the trigger update\_department\_budget updates the department's total payroll and adjusts its budget accordingly.

You can run the SQL queries above to generate reports on employee performance, payroll details, or department budgets.

This setup allows for a comprehensive, dynamic, and responsive employee management system.

To illustrate the scenario, let's work with a set of example data and create the tables for employees, departments, payroll, and performance evaluations.

1. Departments Table:

We'll create a few departments.

CREATE TABLE departments (

department\_id INT PRIMARY KEY AUTO\_INCREMENT,

department\_name VARCHAR(100) NOT NULL,

department\_budget DECIMAL(15, 2) NOT NULL

);

-- Example data for departments

INSERT INTO departments (department\_name, department\_budget) VALUES

('HR', 500000.00),

('IT', 1000000.00),

('Sales', 750000.00);

2. Employees Table:

Now, we’ll create employees and assign them to departments. They will have some initial salary and performance rating.

CREATE TABLE employees (

employee\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

department\_id INT,

salary DECIMAL(15, 2),

hire\_date DATE,

performance\_rating INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Example data for employees

INSERT INTO employees (first\_name, last\_name, department\_id, salary, hire\_date, performance\_rating) VALUES

('John', 'Doe', 1, 55000.00, '2020-01-15', 4),

('Jane', 'Smith', 2, 80000.00, '2019-05-22', 5),

('Alice', 'Brown', 3, 60000.00, '2021-09-10', 3);

3. Payroll Table:

Let’s create the payroll table and populate it with initial payroll records.

CREATE TABLE payroll (

payroll\_id INT PRIMARY KEY AUTO\_INCREMENT,

employee\_id INT,

pay\_date DATE,

salary DECIMAL(15, 2),

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

-- Example data for payroll

INSERT INTO payroll (employee\_id, pay\_date, salary) VALUES

(1, '2023-10-31', 55000.00),

(2, '2023-10-31', 80000.00),

(3, '2023-10-31', 60000.00);

4. Performance Evaluations Table:

We’ll add performance evaluations for employees, including scores and feedback.

CREATE TABLE performance\_evaluations (

evaluation\_id INT PRIMARY KEY AUTO\_INCREMENT,

employee\_id INT,

evaluation\_date DATE,

score INT,

feedback TEXT,

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

-- Example data for performance evaluations

INSERT INTO performance\_evaluations (employee\_id, evaluation\_date, score, feedback) VALUES

(1, '2023-12-01', 4, 'Great teamwork and communication skills.'),

(2, '2023-12-01', 5, 'Outstanding performance and leadership.'),

(3, '2023-12-01', 3, 'Needs improvement in project management skills.');

Step 5: Running Stored Procedures with Example Data

1. Updating Payroll:

For example, if John Doe gets a salary increase, you would call the update\_payroll stored procedure like this:

CALL update\_payroll(1, 60000.00); -- Increase John's salary to 60000

This will update the salary in the employees table and insert a record into the payroll table for John Doe.

2. Adding a Performance Review:

For example, if Jane Smith receives a new performance review, you would call the add\_performance\_review stored procedure:

CALL add\_performance\_review(2, 5, 'Excellent leadership and problem-solving skills.');

This will insert a new performance evaluation record for Jane and update her performance rating.

Step 6: Triggers Example

1. Trigger to Update Department Budget:

Let’s say you increase Alice Brown’s salary to $70,000, and we want the department budget for Sales (department\_id = 3) to be updated automatically:

UPDATE employees SET salary = 70000 WHERE employee\_id = 3;

The trigger update\_department\_budget will automatically update the Sales department budget by recalculating the total payroll for the department.

Step 7: Queries with Example Data

1. Generate Employee Performance Reports:

This query will list employees' performance evaluations, focusing on their most recent review.

SELECT e.employee\_id, e.first\_name, e.last\_name, e.department\_id, e.salary, e.performance\_rating, p.evaluation\_date, p.score, p.feedback

FROM employees e

JOIN performance\_evaluations p ON e.employee\_id = p.employee\_id

WHERE p.evaluation\_date = (

SELECT MAX(evaluation\_date)

FROM performance\_evaluations

WHERE employee\_id = e.employee\_id

)

ORDER BY e.department\_id, e.last\_name;

2. Generate Payroll Summary:

This query will show payroll details for all employees.

SELECT e.employee\_id, e.first\_name, e.last\_name, e.department\_id, p.pay\_date, p.salary

FROM employees e

JOIN payroll p ON e.employee\_id = p.employee\_id

ORDER BY p.pay\_date DESC;

3. Generate Department Budget Report:

This query shows the total payroll for each department and compares it with the department’s current budget.

SELECT d.department\_name, d.department\_budget, SUM(e.salary) AS total\_payroll

FROM departments d

JOIN employees e ON d.department\_id = e.department\_id

GROUP BY d.department\_id;

Summary of Example Data:

Departments: HR, IT, and Sales.

Employees: John Doe, Jane Smith, Alice Brown.

Payroll: Payroll records for the employees with their salaries.

Performance Evaluations: Recent performance evaluations with scores and feedback.

By running these queries and stored procedures, you can manage employee payroll, performance evaluations, department budgets, and generate reports for analysis

Logical, Physical, and Conceptual ER Diagrams

To illustrate the relationships between the entities (tables) and their attributes, we’ll provide the following three types of ER diagrams:

Conceptual ER Diagram (High-level overview)

Logical ER Diagram (With primary keys, foreign keys, and relationships)

Physical ER Diagram (Showing specific details about the data types and the table structure)

1. Conceptual ER Diagram

The Conceptual ER Diagram provides a high-level overview of the entities (tables) and their relationships without focusing on implementation details like primary/foreign keys.

Entities:

Department: Manages employee data for each department.

Employee: Stores employee details.

Payroll: Represents payroll records for employees.

Performance Evaluation: Stores performance reviews for employees.

Relationships:

Department "has" many Employees.

Employee "has" many Payroll Records.

Employee "receives" many Performance Evaluations.

+-----------------+ +----------------+

| Department | | Employee |

+-----------------+ +----------------+

| department\_id PK| | employee\_id PK |

| department\_name | | first\_name |

| department\_budget| | last\_name |

+-----------------+ | salary |

| | hire\_date |

| | performance\_rating |

| | department\_id FK |

| +----------------+

| |

| |

| |

+----------------+ +------------------------+

| Payroll | | Performance Evaluation |

+----------------+ +------------------------+

| payroll\_id PK | | evaluation\_id PK |

| employee\_id FK | | employee\_id FK |

| pay\_date | | evaluation\_date |

| salary | | score |

+----------------+ | feedback |

+------------------------+

2. Logical ER Diagram

The Logical ER Diagram refines the conceptual diagram by specifying primary keys (PK), foreign keys (FK), and the relationships between entities. It also outlines how each entity (table) relates to one another.

Entities:

Department: department\_id, department\_name, department\_budget

Employee: employee\_id, first\_name, last\_name, salary, hire\_date, performance\_rating, department\_id

Payroll: payroll\_id, employee\_id, pay\_date, salary

Performance Evaluation: evaluation\_id, employee\_id, evaluation\_date, score, feedback

+-----------------+ +----------------+

| Department | | Employee |

+-----------------+ +----------------+

| department\_id PK|◄─────┐ | employee\_id PK |

| department\_name | │ | first\_name |

| department\_budget| │ | last\_name |

+-----------------+ │ | salary |

▲ │ | hire\_date |

| │ | performance\_rating |

| │ | department\_id FK (Dept)|

| │ +----------------+

| │

+----------------+ │

| Payroll | │

+----------------+ │

| payroll\_id PK | │

| employee\_id FK |───────┘

| pay\_date |

| salary |

+----------------+

|

|

+------------------------+

| Performance Evaluation |

+------------------------+

| evaluation\_id PK |

| employee\_id FK |

| evaluation\_date |

| score |

| feedback |

+------------------------+

Relationships:

One Department has many Employees (1-to-many relationship).

One Employee has many Payroll records (1-to-many relationship).

One Employee has many Performance Evaluations (1-to-many relationship).

3. Physical ER Diagram

The Physical ER Diagram gives a more detailed view of how the data will be stored in the database. It includes specific data types for each column and outlines primary and foreign key constraints.

+------------------------------+

| Department |

+------------------------------+

| department\_id INT PK |

| department\_name VARCHAR(100) |

| department\_budget DECIMAL(15, 2)|

+------------------------------+

▲

|

+------------------------------+

| Employee |

+------------------------------+

| employee\_id INT PK |

| first\_name VARCHAR(50) |

| last\_name VARCHAR(50) |

| salary DECIMAL(15, 2) |

| hire\_date DATE |

| performance\_rating INT |

| department\_id INT FK |

+------------------------------+

│

+------------------------------+

| Payroll |

+------------------------------+

| payroll\_id INT PK |

| employee\_id INT FK |

| pay\_date DATE |

| salary DECIMAL(15, 2) |

+------------------------------+

│

+------------------------------+

| Performance Evaluation |

+------------------------------+

| evaluation\_id INT PK |

| employee\_id INT FK |

| evaluation\_date DATE |

| score INT |

| feedback TEXT |

+------------------------------+

Explanation of Data Types:

INT: Integer (e.g., employee\_id, department\_id, evaluation\_id).

VARCHAR: Variable-length string (e.g., department\_name, first\_name, last\_name).

DECIMAL(15,2): Decimal numbers with up to 15 digits and 2 decimal places (e.g., salary, department\_budget).

DATE: Date type (e.g., hire\_date, pay\_date, evaluation\_date).

TEXT: Long text (e.g., feedback).

Key Points:

The conceptual diagram shows the broad relationships between entities.

The logical diagram adds the keys and specific relationships (foreign key links).

The physical diagram is the implementation level where we define actual column types and constraints to create the tables in the database.

This approach will help you visualize the structure of your system at various levels of abstraction, ensuring a solid understanding of both the relationships and technical details required for building the database.