Caroline Syombua Nzioka
Employee Management System Project

Table Of Contents

Project Overview:

This project is an employee management system where the administrator can add new employees, a new department and can also delete and edit an employee through a website. The database is a relational database that has 10 tables that are linked through foreign key relationships.

- ☐ **Database Implementation**: ProsgreSQL was used as the SQL language. The GUI used is PGAdmin4.
- ☐ **Data Insertion:** Used fake data with SQL insert statements to populate the tables with data.
- ☐ **Database Integration:** Created a website using React, Node.js and Express.js.
  React is a JavaScript library that was used to build the user interface for the front end.
  Node was used for the server side and installed Pg to connect the payroll database to the user interface.
  React interacts with Node.js which then communicates with ProsgreSQL.

- ☐ **Database tables:**

1- Employee: Stores employee information and is used for managing and tracking all the employees.

Attributes:

emp_ID: Unique identifier for each employee.
manager_ID: Foreign key linking to the employee table. (A manager is still an employee)
address_ID: Foreign key linking to the address table.
paygrade_ID: Foreign key linking to the paygrade table.
department_ID: Foreign key linking to the Department table.
first_name: First name of the employee.
middle_name: Middle name of the employee.
last_name: Last name of the employee.
email: Email address of the employee.
phone_number: Phone number of the employee.
Ssn: Social security number of the employee.
birth_date: Date of birth of the employee.
Username: username assigned to the employee to log into the website.
Password_: Password of the employee.
position_name: Current Position or job title of the employee.

2- Department: Used to store all the current departments within the organization

Attributes:

department_ID: Unique identifier for each department.

Department_name: Name of the department.

3- Admin: Used for managing access to the HR system, the management website

Attributes:

admin_ID: Unique identifier for each admin.
emp_ID: Employee ID of the admin (only two can access, Caroline and Mark).
email: Email address of the admin, used to log into the website.
password_: Password of the admin, used to log into the website.

4- Addresses: Used to store all the employee address

Attributes:

address_ID: Unique identifier for each address.
address_line_one: First line that holds an address.
Address_line_two: Second line that holds the address.
city: The city of the address.
state: The state of the address.
zip: The zip of the address.
country: The country of the address.

5- Deduction: Used for calculating and deducting the taxes for Pennsylvania.

Attributes:

deduction_ID: Unique identifier for each deduction row.
federal_WH: Federal withholding rate.
medicare: Medicare withholding rate.
social_security: Social security withholding rate.
penn_state_WH: Pennsylvania state withholding rate.
township_tax: Township tax withholding rate.
pennsylvania_ULHCWF: Pennsylvania Unemployment Compensation Workers' Compensation Fund rate.
total_deduction: Total deduction amount after adding all the deductions.

6- Paygrade: Used for calculating the employee salary and net pay depending on the level of paygrade they are in. employees in different levels receive a different net salary.

Attributes:

paygrade_ID: Unique identifier for each paygrade record.
paygrade: Different levels of paygrade.
pay_rate: The hourly pay rate for the employee.
overtime_rate: The overtime pay rate for the employee.

7- Salary: Used for calculating the base pay which takes into account the employees time worked and overtime. This table focuses on ensuring that the employees are paid for their overtime.

Attributes:

salary_ID: Unique identifier for each salary record.
emp_ID: Employee ID of the employee.
paygrade_ID: Paygrade ID corresponds to the employee's pay grade level.
time_punch_ID: Time punch ID corresponds to the employee's time worked.
base_salary: is the payrate from paygrade table * (total_hours - overtime_hours) from time_punch table.
overtime_pay: overtime_rate from paygrade table * overtime_hours.
gross_salary: base_salary plus overtime_pay.

8- Payroll: Used for calculating the take home pay for the employees. This includes taking the gross salary and deducting the total deductions to get the final net pay that will be paid out to the employees.

Attributes:

payroll_ID: Unique identifier for each payroll record.
emp_ID: Employee ID of the employee, referencing the employee table.
salary_ID: Salary ID referencing the salary table.
deduction_ID: Deduction ID referencing the deduction table.
net_salary: Net salary amount after deductions which is gross salary from salary table minus total_deductions from the deduction table.

9- time_off: Used to store time off requests requested by the employee.

Attributes:

time_req_ID: Unique identifier for each time off record.
emp_ID: Employee ID of the employee requesting time off, which references the employee table.
start_date: Start date of the time off.
end_date: End date of the time off.
comments_: Comments or reason for requesting the time off.
status: The status of the time off request (Approved, Pending).

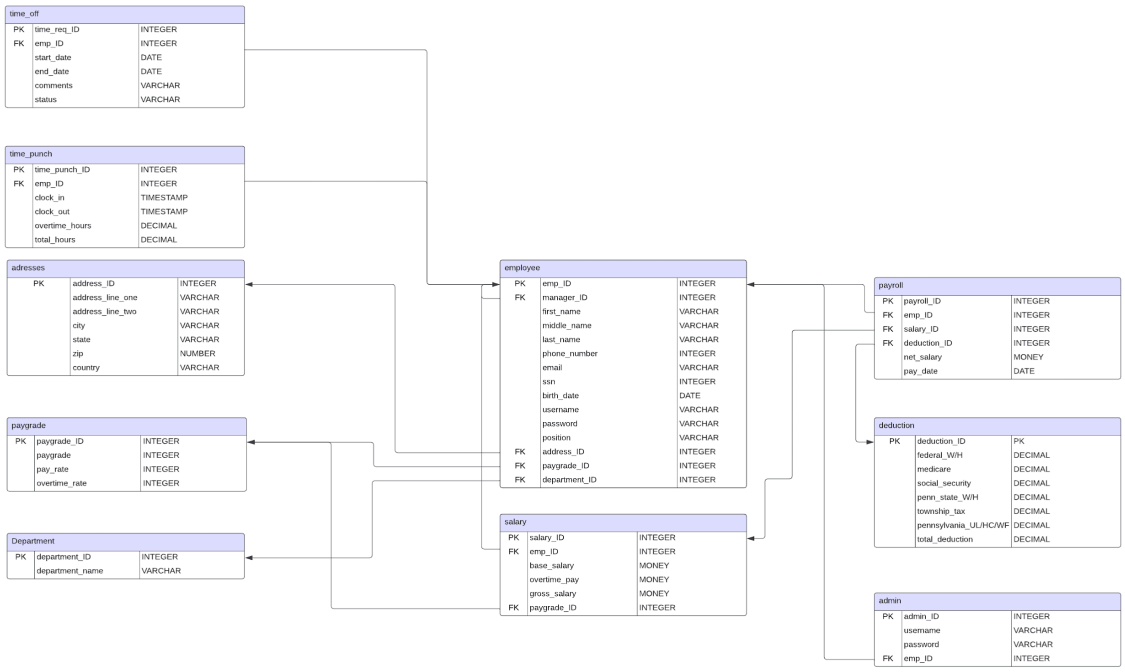
10- time_punch: Used to track the employee hours. Records clock in and clock out punches.

Attributes:

time_punch_ID: A Unique identifier for each time punch record.

emp_ID: Employee ID of the employee which references the employee table.
clock_in: The date and time that the employee clocked in.
clock_out: The date and time that the employee clocked out.
total_hours: The total hours worked for that pay period.
overtime_hours: The overtime hours worked for that pay period.

Relational Database Schema Diagram



Triggers

- **1- This trigger calculated the total hours worked and the overtime hours worked for each employee based on the clock in and clock out time punches. This trigger inserts the total hours and overtime hours in the time punch table.**

```
CREATE OR REPLACE FUNCTION calculate_hours()
        RETURNS TRIGGER
        LANGUAGE plpgsql
        AS $$
                BEGIN
                NEW.total_hours := EXTRACT(EPOCH FROM (NEW.clock_out - NEW.clock_in))
/ 3600.0;
        IF NEW.total_hours > 8.0 THEN
                NEW.overtime_hours := NEW.total_hours - 8.0;
        ELSE
                 NEW.overtime_hours := 0.0;
        END IF;
   RETURN NEW;
END;
$$;

CREATE TRIGGER calculate_hours_trigger
BEFORE INSERT ON time_punch
FOR EACH ROW
EXECUTE FUNCTION calculate_hours();
```

- **2- This trigger calculates the base salary, overtime pay and gross salary in the salary table when a new row is inserted in the salary table.**

```
 CREATE OR REPLACE FUNCTION salary_calculation()
        RETURNS TRIGGER
        LANGUAGE plpgsql
        AS
        $$
                DECLARE
                base_salary MONEY;
                overtime_pay MONEY;
                gross_salary MONEY;
                BEGIN
                        SELECT pg.pay_rate * (tp.total_hours - tp.overtime_hours),
```

```
        pg.overtime_rate * tp.overtime_hours
                INTO base_salary, overtime_pay
    FROM
        paygrade pg
        JOIN employee e ON e.paygrade_ID = pg.paygrade_ID
        JOIN time_punch tp ON tp.emp_ID = e.emp_ID
    WHERE
        e.emp_ID = NEW.emp_ID
        AND tp.time_punch_ID = NEW.time_punch_ID;

    gross_salary = base_salary + overtime_pay;

    NEW.base_salary = base_salary;
    NEW.overtime_pay = overtime_pay;
    NEW.gross_salary = gross_salary;

    RETURN NEW;
        END;
$$;

CREATE TRIGGER calculate_salary
BEFORE INSERT ON salary
FOR EACH ROW
EXECUTE FUNCTION salary_calculation();
```

Queries:

1-This shows the total hours worked by each employee showing their departments.

```
SELECT concat (e.first_name, ' ', e.middle_name, ' ', e.last_name) AS employee_Name,
SUM(tp.total_hours) AS total_hours_worked, e.position_name, d.department_name
FROM employee e
JOIN time_punch tp ON e.emp_ID = tp.emp_ID
JOIN department d ON e.department_ID = d.department_ID
GROUP BY employee_name, e.position_name, d.department_name
ORDER BY total_hours_worked DESC;
```

2- which employee gets the most money from the company.

```
SELECT e.emp_ID, d.department_name, p.net_salary, concat (e.first_name, ' ',
e.middle_name, ' ', e.last_name) AS employee_Name
FROM employee e
JOIN department d ON e.department_ID = d.department_ID
JOIN payroll p ON e.emp_ID = p.emp_ID
WHERE p.net_salary = (SELECT MAX(net_salary) FROM payroll);
```

3- The average Accounting salary

```
SELECT AVG (p.net_salary::numeric) AS Average_accounting_salary
FROM payroll p
JOIN employee e ON p.emp_ID = e.emp_ID
JOIN department d ON e.department_ID = d.department_ID
WHERE d.department_name = 'Accounting';
```

4- Payroll Calculations calculating the gross salary, deductions and net salary of all employees.

```
SELECT e.emp_ID, concat (e.first_name, ' ',e.middle_name, ' ', e.last_name) AS
employee_name,
(pg.pay_rate * 40) AS base_salary,
(pg.overtime_rate * t.overtime_hours) AS overtime_pay,
(pg.pay_rate * 40) + (pg.overtime_rate * t.overtime_hours) AS Gross_salary,
(((pg.pay_rate * 40) + (pg.overtime_rate * t.overtime_hours)) * d.total_deduction) AS
Total_deduction,
(pg.pay_rate * 40) + (pg.overtime_rate * t.overtime_hours) - (((pg.pay_rate * 40) +
(pg.overtime_rate * t.overtime_hours)) * d.total_deduction) AS Net_salary
FROM employee e
JOIN paygrade pg on e.paygrade_ID = pg.paygrade_ID
JOIN time_punch t on t.emp_ID = e.emp_ID
CROSS JOIN deduction d
```

5- Looks at which employees do not come on time. clock-in time is past 9:15 AM. (Normal work hours is 9-5)

```
SELECT t.emp_ID, concat (e.first_name, ' ',e.middle_name, ' ', e.last_name) AS
employee_name, t.clock_in
FROM time_punch t
JOIN employee e on e.emp_ID = t.emp_ID
WHERE EXTRACT(HOUR FROM clock_in) > 9 OR (EXTRACT(HOUR FROM clock_in) = 9
AND EXTRACT(MINUTE FROM clock_in) > 15);
```

6- Finding the average salary in each department

```
SELECT d.department_name, AVG (p.net_salary::numeric) AS avg_salary
FROM payroll p
JOIN employee e ON p.emp_ID = e.emp_ID
JOIN department d ON e.department_ID = d.department_ID
GROUP BY department_name;
```

7- Look at employees that are likely to receive a promotion- They come early and leave late. The manager would have to assess their performance at work before decisions are made.

```
SELECT t.emp_ID, d.department_name, concat (e.first_name, ' ',e.middle_name, ' ',
e.last_name) AS employee_name, t.clock_in, t.clock_out
FROM time_punch t
JOIN employee e on e.emp_ID = t.emp_ID
JOIN department d ON e.department_ID = d.department_ID
WHERE EXTRACT(HOUR FROM clock_in) = 8 AND EXTRACT(HOUR FROM clock_out) > 17;
```

8- Look at which employees take the most leave and why.

```
SELECT e.emp_ID, concat (e.first_name, ' ',e.middle_name, ' ', e.last_name) AS
employee_name,
COUNT(t.time_req_ID) AS total_leave_requests, t.comments_
FROM employee e
JOIN time_off t ON e.emp_ID = t.emp_ID
GROUP BY e.emp_ID, employee_name, t.comments_
ORDER BY total_leave_requests DESC;
```

9- This looks at what department makes the most overtime

```
SELECT d.department_name, sum(t.overtime_hours) AS total_overtime
FROM department d
JOIN employee e on d.department_ID = e.department_ID
JOIN time_punch t ON e.emp_ID = t.emp_ID
GROUP BY department_name
```

10- Which employee makes the most overtime:

```
SELECT d.department_name, concat (e.first_name, ' ',e.middle_name, ' ', e.last_name) AS
employee_name, sum(t.overtime_hours) AS total_overtime
FROM department d
JOIN employee e on d.department_ID = e.department_ID
JOIN time_punch t ON e.emp_ID = t.emp_ID
GROUP BY e.emp_ID, d.department_name
ORDER BY d.department_name
```

11- How much cost is spent on overtime:

```
SELECT e.emp_ID, d.department_name, sum(s.overtime_pay) AS total_overtime_pay, concat
(e.first_name, ' ', e.middle_name, ' ', e.last_name) AS employee_Name
FROM employee e
JOIN department d ON e.department_ID = d.department_ID
JOIN salary s ON e.emp_ID = s.emp_ID
GROUP BY e.emp_ID, d.department_name
ORDER BY total_overtime_pay DESC
```

12- Looking at which employees make above average (based on the whole company average)

```
SELECT concat (e.first_name, ' ',e.middle_name, ' ', e.last_name) AS employee_name,
p.net_salary, (SELECT AVG(net_salary::numeric)
```

```
FROM payroll) AS Company_average
FROM employee e
JOIN payroll p ON e.emp_ID = p.emp_ID
WHERE p.net_salary::numeric > ( SELECT AVG (net_salary::numeric)
                                FROM payroll);
```

13- How many employees from each department

```
SELECT department_name, COUNT(*) AS total_employees
FROM employee
JOIN department ON employee.department_ID = department.department_ID
GROUP BY department_name;
```

14- Finding the net salaries of the employees

```
SELECT p.net_salary, concat (e.first_name, ' ',e.middle_name, ' ', e.last_name) AS
employee_name
FROM payroll p
JOIN employee e on p.emp_ID = e.emp_ID;
```

15- The average gross salary from each paygrade

```
SELECT p.paygrade, AVG(s.gross_salary::numeric) AS avg_gross_salary
FROM salary s
JOIN paygrade p ON s.paygrade_ID = p.paygrade_ID
GROUP BY p.paygrade
ORDER BY avg_gross_salary DESC;
```