COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

Databases and Database Systems - COMP3030

Group Project Design Document

Members

Student name: Nguyen Quoc Dang - V202200767

Student name: Le Nguyen Gia Binh - V202200

Student name: Thai Ba Hung - V202200

Instructors

Dr. Le Duy Dung

Date of Submission

20 May 2025

VINUNIVERSITY

I Conceptual & Logical Design

1 Functional & Non-functional Requirements

1.1 Functional Requirements:

- Login Function: Login function for different user
- Personal View: View personal information (Payroll Information, Personal Performance Review, Feedback, etc)
- Employee Management: Add, update, and delete employee records.
- Attendance Tracking: Track employee attendance, including leave requests and working hours.
- Payroll Management: Calculate and generate monthly payroll for employees.
- Performance Evaluation: Store and manage performance reviews, feedback, and appraisal data.
- Role-Based Access Control: Different user roles (HR manager, employee, admin) with specific permissions.

1.2 Non-functional Requirements:

- Scalability: System should be able to handle a growing number of employees.
- Security: Secure access to sensitive employee data with proper encryption and authentication.
- Reliability: System must ensure high availability with minimal downtime.

2 Entity–Relationship Diagram

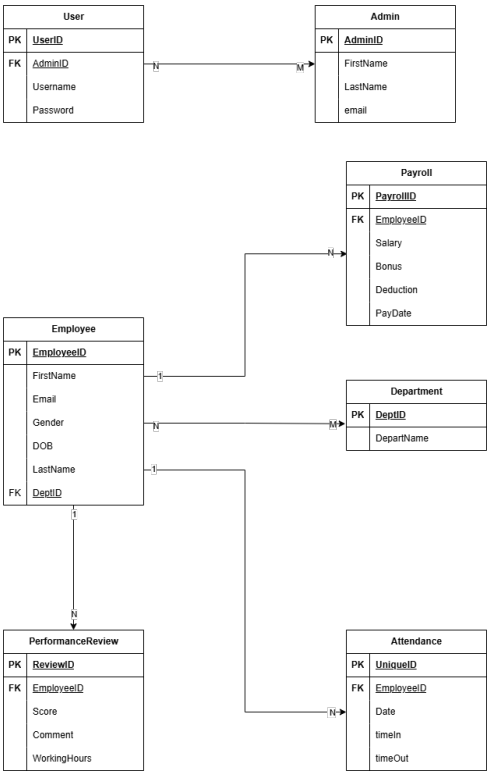

Figure 1: Entity-Relationship Diagram

In all tables:

- All attributes depend only on the primary key (1NF).

- No partial dependency on composite keys (2NF).
- No transitive dependencies (3NF).

Therefore, all tables are normalized up to **3NF**.

II Physical Schema Definition

1 DDL scripts

```
-- Department Table
CREATE TABLE Department (
    DepartmentID INT AUTO_INCREMENT PRIMARY KEY,
    DeptName VARCHAR(100) NOT NULL UNIQUE
);

-- Employee Table
CREATE TABLE Employee (
    EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DOB DATE,
    Phone VARCHAR(15),
    Email VARCHAR(100) UNIQUE,
    Gender binary,
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);

-- Attendance Table
CREATE TABLE Attendance (
    AttendanceID INT AUTO_INCREMENT PRIMARY KEY,
    EmployeeID INT NOT NULL,
    Date DATE NOT NULL,
    timeIn TIME,
    timeOut TIME,
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
    UNIQUE(EmployeeID, Date)
);

-- Payroll Table
CREATE TABLE Payroll (
    PayrollID INT AUTO_INCREMENT PRIMARY KEY,
    EmployeeID INT NOT NULL,
    Salary DECIMAL(15,2) NOT NULL,
    Bonus DECIMAL(15,2) DEFAULT 0,
    Deduction DECIMAL(15,2) DEFAULT 0,
    PayDate DATE NOT NULL,
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);

-- Performance Review Table
CREATE TABLE PerformanceReview (
    ReviewID INT AUTO_INCREMENT PRIMARY KEY,
    EmployeeID INT NOT NULL,
    ReviewDate DATE NOT NULL,
```

```
    Score INT CHECK (Score BETWEEN 1 AND 10),
    Comments TEXT,
    WorkingHours INT NOT NULL,
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);

-- Admin Table
CREATE TABLE `Admin` (
    AdminID INT AUTO_INCREMENT primary key,
    FirstName varchar(50),
    LastName varchar(50),
    email varchar(50)
);

-- User Table
CREATE TABLE UserAccount (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    adminID INT NOT NULL UNIQUE,
    Username VARCHAR(50) NOT NULL UNIQUE,
    `password` VARCHAR(255) NOT NULL,
    FOREIGN KEY (adminID) REFERENCES `Admin`(adminID)
);
```

2 Definitions of views, indexes, and partitioning strategy

2.1 Views

Views provide virtual tables built from queries, simplifying complex joins and giving convenient access to data.

- View for Employee Payroll Summary:

```
CREATE VIEW EmployeePayrollSummary AS
SELECT
    e.EmployeeID,
    CONCAT(e.FirstName, ' ', e.LastName) AS FullName,
    p.PayDate,
    p.Salary,
    p.Bonus,
    p.Deductions,
    (p.Salary + p.Bonus - p.Deductions) AS NetPay
FROM Employee e
JOIN Payroll p ON e.EmployeeID = p.EmployeeID;
```

- View: Latest Performance Review per Employee:

```
CREATE VIEW LatestPerformanceReview AS
SELECT
    pr.EmployeeID,
    CONCAT(e.FirstName, ' ', e.LastName) AS FullName,
    pr.ReviewDate,
    pr.Score,
    pr.Comments
FROM PerformanceReview pr
JOIN Employee e ON pr.EmployeeID = e.EmployeeID
WHERE (pr.EmployeeID, pr.ReviewDate) IN (
    SELECT EmployeeID, MAX(ReviewDate)
```

```
FROM PerformanceReview
GROUP BY EmployeeID);
```

2.2 Indexes

Indexes speed up data retrieval for frequently queried columns. We will create a search bar to retrieve the employee with their names or emails. To make the searching more efficient, we will use a B-tree Indexes for the Employee tables.

III Task Division & Project Plan

1 Breakdown of team member responsibilities

- **Team Lead: Le Nguyen Gia Binh**
  - Oversees the entire project
  - Coordinates the team's activities
  - Ensures deadlines and milestones are met
- **Backend Developer: Thai Ba Hung**
  - Designs the database structure
  - Develops APIs for the application
  - Implements backend logic
- **Frontend Developer: Le Nguyen Gia Binh**
  - Designs and implements the user interface (UI)
  - Ensures a smooth user experience (if UI is applicable)
- **Database Administrator: Nguyen Quoc Dang**
  - Sets up the database environment
  - Designs and manages the database schema
  - Maintains database performance and security

2 Timeline/Gantt chart showing deliverables for Activities 1–5

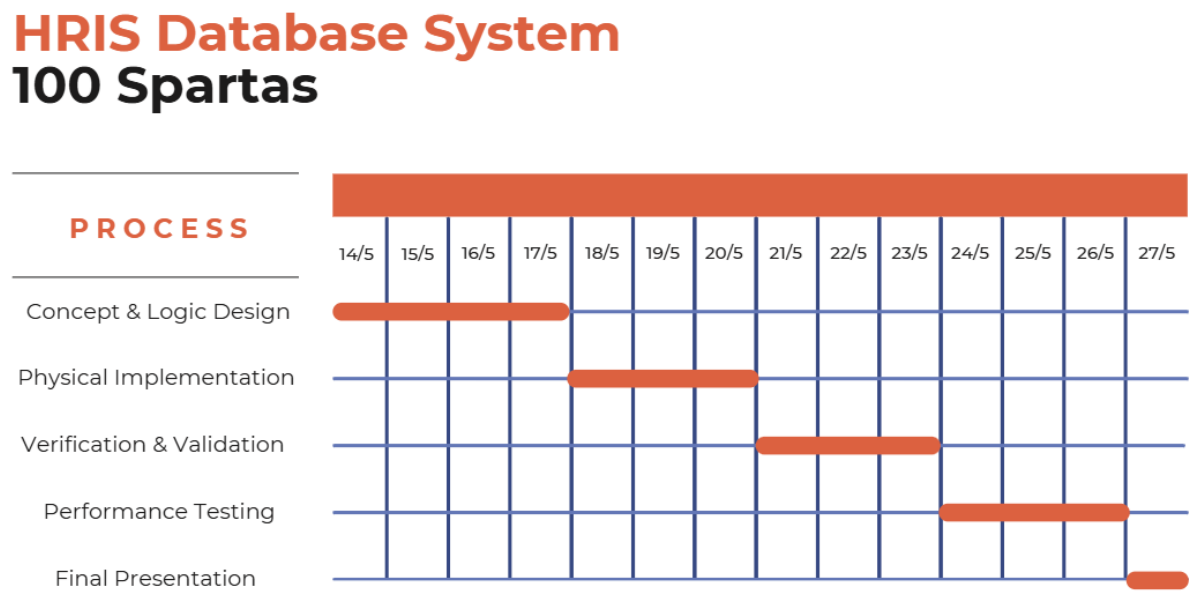

Figure 2: Project Timeline Gantt Chart

IV. Supporting Documentation

1. Brief Rationale for Design Decisions

The database design ensures data integrity and efficiency through normalization up to Third Normal Form (3NF), which eliminates partial and transitive dependencies. The use of foreign keys maintains referential integrity between related tables, such as `Employee` and `Department`, or `Payroll` and `Employee`. Indexes are implemented on frequently searched columns like employee names and emails to optimize query performance. Views are defined to simplify complex queries, such as employee payroll summaries and latest performance reviews, improving ease of access for common reports.

2. Preliminary Notes on Sample Data Loading Approach

Sample data will be loaded in stages, starting with the `Department` and `Admin` tables, followed by `Employee` and related entities like `Attendance`, `Payroll`, and `PerformanceReview`. This order respects foreign key dependencies and prevents referential integrity errors. Bulk data insertion scripts will be utilized to efficiently populate large datasets. Data validation and constraint checks will be enforced during loading to maintain consistency.

3. Project Repository

For detailed project documentation, code, and scripts, please refer to our GitHub repository:

VinUni Database Course Final Project Repository