**Employee Management System**

This will help set up a database for the Employee Management System (EMS) to manage the following functionalities:

1. **Employee Details**:
   * Store and manage information about individual employees, including their names, addresses, genders, and salary details.
2. **Department Details**:
   * Maintain records of different departments within the organization, including their names and locations.
3. **Project Details**:
   * Track information related to various projects in the organization, including project names, budgets, and locations.
4. **Salary Records**:
   * Manage salary records for employees, including salary amounts, start dates, and end dates.
5. **Attendance Records**:
   * Keep track of employee attendance records, including attendance dates and statuses (e.g., Present, Absent).

We will use MySQL Workbench as the DBMS to create the database and its related operations.

1. **Introduction to MySQL Workbench**

What is MySQL Workbench?

MySQL Workbench is a unified cross-platform, open-source relational database design tool that adds functionality and ease to MySQL and SQL development work. MySQL Workbench provides data modeling, SQL development, and various administration tools for configuration. It also offers a graphical interface to work with the databases in a structured way.

* You can create a Graphical Model using MySQL Workbench
* MySQL Workbench provides reverse engineering for live databases to models
* MySQL Workbench offers a forward engineering model to a script/live database

1. **Installation of MySQL Workbench**

The installation process is similar to other operating systems.

1. Open the MySQL website on a browser. Click on the following link: [MySQL Downloads](https://www.mysql.com/downloads/).
2. Select the Downloads option.
3. Select MySQL Installer for Windows.
4. Choose the desired installer and click on download.
5. After it downloads the installer, open it.
6. It will ask for permission; when it does, click Yes. The installer will then open. Now, it will ask to choose the setup type, here, select Custom.
7. Click on Next. With this, you will be able to install MySQL server, MySQL Workbench, and MySQL shell.
8. Open MySQL Servers, select the server you want to install, and move it to the  Products/Features to be installed window section. Now, expand Applications, choose MySQL Workbench and MySQL shell. Move both of them to ‘Products/Features to be installed’.
9. Click on the Next button. Now, click on the Execute button to download and install the MySQL server, MySQL Workbench, and the MySQL shell.
10. Once the product is ready to configure, click on Next. Under Type and Networking, go with the default settings and select Next.
11. For authentication, use the recommended strong password encryption.
12. Set your MySQL Root password and click on next.
13. Go for the default windows service settings and under apply configuration, click on execute. Once the configuration is complete, click on finish.
14. Complete the installation. This will now launch the MySQL Workbench and the MySQL Shell.

Once MySQL Workbench is installed, select the Local instance and enter the password.

Now, you can use the MySQL query tab to write your SQL queries.

**Identify Entities**

* Start by identifying the main entities in your system. These are the objects or concepts about which you want to store data.
* Each entity should correspond to a table in your database.

**Define Attributes**

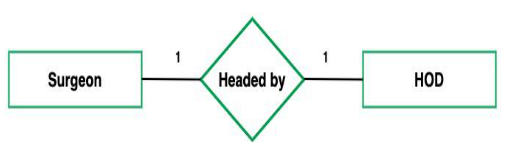
* For each entity, list the attributes (properties or fields) that describe it.
* These attributes will become columns in the corresponding database table.

**Identify Relationships**

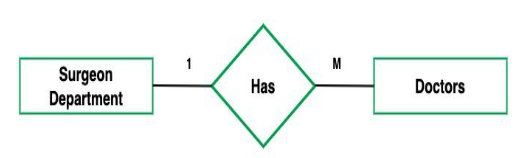
* Determine how entities are related to each other. There are three types of relationships: one-to-one (1:1), one-to-many (1:N), and many-to-many (N:M).
* Represent these relationships using lines connecting the entities.

Let’s see a few examples of relationships:

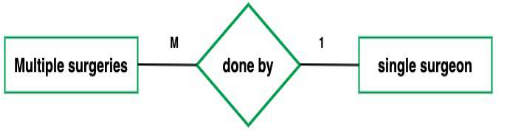
1. **One-to-One:** When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.



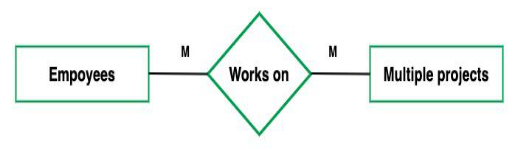
1. **One-to-Many:** In one-to-many mapping as well where each entity can be related to more than one relationship and the total number of tables that can be used in this is 2. Let us assume that one surgeon deparment can accomodate many doctors. So the Cardinality will be 1 to M. It means one deparment has many Doctors.



**3. Many-to-One:** When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students.



**Many-to-Many:**When entities in all entity sets can take part more than once in the relationship cardinality is many to many.



**Cardinality Notation**

Cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship.

* Use notation (such as Crow's Foot Notation or Chen Notation) to indicate the cardinality of each relationship.
* Cardinality describes how many instances of one entity are related to how many instances of another entity.
* Common notations include:
* One (1)
* Zero or one (0..1)
* Many (N)
* Zero or many (0..N)

**Optional:**

**Add Attributes and Constraints**

* Include additional information in your ERD, such as primary keys, foreign keys, and constraints (e.g., unique constraints).

**Create the Diagram**

* Use specialized diagramming software or tools (e.g., Lucidchart, draw.io, or even pen and paper) to create your ERD.

**Refine and Review:**

* Review your ERD with stakeholders and team members to ensure it accurately represents the data model and relationships. Make any necessary refinements.
* Entities of Employee Management System

1. Employee
2. Department
3. Salary
4. Project
5. Attendance

* let’s identify the attributes and relationships of each entity for the Employee Management System.
* Employee

Attributes:

* **EmployeeID** (Primary Key): A unique identifier for each employee.
* **First\_name**: The first name of the employee.
* **Last\_name**: The last name of the employee.
* **Address**: The address of the employee.
* **Salary**: The salary of the employee.
* **Sex**: The gender of the employee.
* **Date\_of\_Birth**: The date of birth of the employee.
* **Email**: The email address of the employee.
* **Phone**: The phone number of the employee.
* **DepartmentID** (Foreign Key): The department to which the employee belongs.
* **ManagerID** (Self-Referencing Foreign Key): The manager of the employee, referring to another employee in the same table.

**Relationship:**

Many-to-One with Department: Each employee belongs to one department.

* Department

Attributes:

* **DepartmentID** (Primary Key): A unique identifier for each department.
* **Name**: The name of the department.
* **Location**: The location or office of the department.

**Relationship**:

One-to-Many with Employee: Each department can have multiple employees.

* Project

Attributes

* **ProjectID** (Primary Key): A unique identifier for each project.
* **Name**: The name of the project.
* **Description**: A description of the project.
* **StartDate**: The start date of the project.
* **EndDate**: The end date of the project.
* **Salary**

Attributes:

* **SalaryID** (Primary Key): A unique identifier for each salary record.
* **EmployeeID** (Foreign Key): The employee to whom the salary record belongs.
* **SalaryAmount**: The amount of the salary.
* **StartDate**: The start date of the salary period.
* **EndDate**: The end date of the salary period.

Relationship:

Many-to-One with Employee: Each salary record belongs to one employee.

* **Attendance**

Attributes:

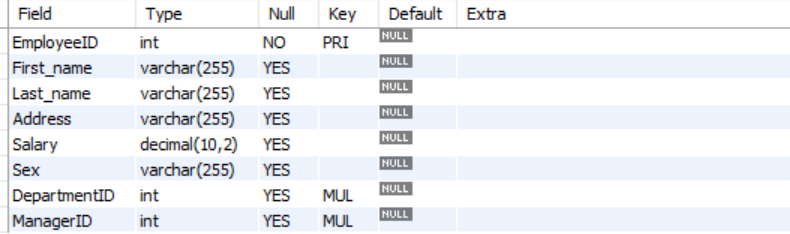
* **AttendanceID** (Primary Key): A unique identifier for each attendance record.
* **EmployeeID** (Foreign Key): The employee to whom the attendance record belongs.
* **AttendanceDate**: The date of the attendance record.
* **Status**: The status of attendance (e.g., Present, Absent).

**Relationship**:

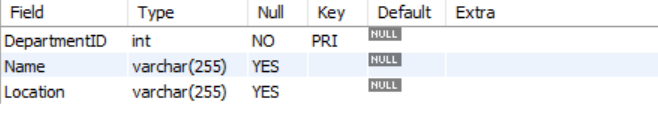
Many-to-One with Employee: Each attendance record is associated with one employee.

**Table Structure**

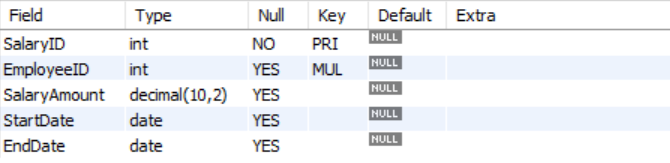
1. **Employee**



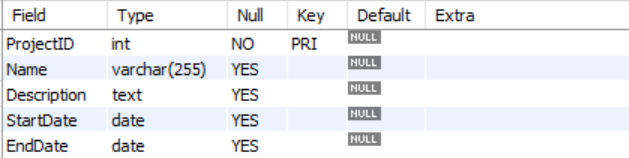
1. **Department**



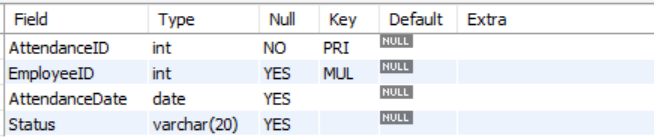
1. **Salary**



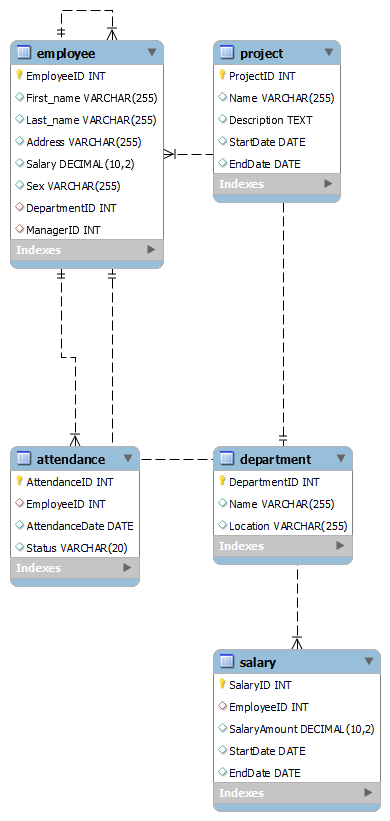
1. **Project**



1. **Attendance**



* ERD Diagram



**In this ERD:**

* **Employee - Department Relationship**:
  + The Employee table has a Many-to-One relationship with the Department table.
  + This means that each employee belongs to one department, and each department can have multiple employees.
  + It is represented by the foreign key attribute **DepartmentID** in the Employee table, which references the primary key **DepartmentID** in the Department table.
* **Employee - Manager Relationship**:
  + The Employee table has a Self-Referencing relationship for representing managers and their subordinates.
  + An employee can have another employee (manager) as their manager.
  + It is represented by the **ManagerID** attribute in the Employee table, which references the **EmployeeID** (self-referencing) in the same Employee table.
* **Salary - Employee Relationship**:
  + The Salary table has a Many-to-One relationship with the Employee table.
  + This means that each salary record is associated with one employee, but an employee can have multiple salary records (e.g., for different time periods).
  + It is represented by the foreign key attribute **EmployeeID** in the Salary table, which references the primary key **EmployeeID** in the Employee table.
* **Attendance - Employee Relationship**:
  + The Attendance table has a Many-to-One relationship with the Employee table.
  + Each attendance record is associated with one employee, but an employee can have multiple attendance records (e.g., for different dates).
  + It is represented by the foreign key attribute **EmployeeID** in the Attendance table, which references the primary key **EmployeeID** in the Employee table.
* **Creating a Database**

Using MySQL Workbech, create a new database for your Employee management system.

Query:

create database employeedata;

* Using a database

Query:

use employeedb;

* Creating the Tables for each entity

Query:

CREATE TABLE Employee (

EmployeeID INT PRIMARY KEY,

First\_name VARCHAR(255),

Last\_name VARCHAR(255),

Address VARCHAR(255),

Salary DECIMAL(10, 2),

Sex VARCHAR(255),

DepartmentID INT,

ManagerID INT,

FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID),

FOREIGN KEY (ManagerID) REFERENCES Employee(EmployeeID)

);

CREATE TABLE Department (

DepartmentID INT PRIMARY KEY,

Name VARCHAR(255),

Locat ion VARCHAR(255)

);

CREATE TABLE Project (

ProjectID INT PRIMARY KEY,

Name VARCHAR(255),

Description TEXT,

StartDate DATE,

EndDate DATE

);

CREATE TABLE Salary (

SalaryID INT PRIMARY KEY,

EmployeeID INT,

SalaryAmount DECIMAL(10, 2),

StartDate DATE,

EndDate DATE,

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

);

CREATE TABLE Attendance (

AttendanceID INT PRIMARY KEY,

EmployeeID INT,

AttendanceDate DATE,

Status VARCHAR(20),

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

);

* **Insert records**

INSERT INTO Employee (EmployeeID, First\_name, Last\_name, Address, Salary, Sex, DepartmentID, ManagerID)

VALUES

(1, 'John', 'Doe', '123 Main St', 60000, 'Male', 1, NULL),

(2, 'Jane', 'Smith', '456 Elm St', 75000, 'Female', 1, 1),

(3, 'Bob', 'Johnson', '789 Oak St', 80000, 'Male', 2, 1),

(4, 'Alice', 'Williams', '101 Pine St', 90000, 'Female', 2, 1);

INSERT INTO Department (DepartmentID, Name, Location)

VALUES

(1, 'HR Department', 'New York'),

(2, 'Engineering Department', 'San Francisco'),

(3, 'Sales Department', 'Chicago');

INSERT INTO Project (ProjectID, Name, Description, StartDate, EndDate)

VALUES

(1, 'Project A', 'Software Development', '2023-01-15', '2023-07-30'),

(2, 'Project B', 'Product Enhancement', '2023-02-10', '2023-06-20'),

(3, 'Project C', 'Sales Expansion', '2023-03-05', '2023-08-15');

INSERT INTO Salary (SalaryID, EmployeeID, SalaryAmount, StartDate, EndDate)

VALUES

(1, 1, 60000, '2023-01-01', '2023-12-31'),

(2, 2, 75000, '2023-01-01', '2023-12-31'),

(3, 3, 80000, '2023-01-01', '2023-12-31'),

(4, 4, 90000, '2023-01-01', '2023-12-31');

INSERT INTO Attendance (AttendanceID, EmployeeID, AttendanceDate, Status)

VALUES

(1, 1, '2023-01-01', 'Present'),

(2, 2, '2023-01-01', 'Present'),

(3, 3, '2023-01-01', 'Absent'),

(4, 4, '2023-01-01', 'Present');

* Select records

Retrieve all employee details

SELECT \* FROM Employee;

-- Retrieve specific employee details by EmployeeID (e.g., EmployeeID = 1)

SELECT \* FROM Employee WHERE EmployeeID = 1;

-- Retrieve employees in a specific department (e.g., DepartmentID = 1)

SELECT \* FROM Employee WHERE DepartmentID = 1;

* Update records

-- Update the salary of an employee (e.g., EmployeeID = 1)

UPDATE Employee

SET Salary = 65000

WHERE EmployeeID = 1;

-- Update the manager for an employee (e.g., EmployeeID = 2)

UPDATE Employee

SET ManagerID = 3

WHERE EmployeeID = 2;

-- Update the location of a department (e.g., DepartmentID = 1)

UPDATE Department

SET Location = 'Los Angeles'

WHERE DepartmentID = 1;

* Delete Queries

-- Delete an employee by EmployeeID (e.g., EmployeeID = 1)

DELETE FROM Employee

WHERE EmployeeID = 1;

-- Delete a department by DepartmentID (e.g., DepartmentID = 2)

DELETE FROM Department

WHERE DepartmentID = 2;

-- Delete a project by ProjectID (e.g., ProjectID = 3)

DELETE FROM Project

WHERE ProjectID = 3;

-- Delete an attendance record by AttendanceID (e.g., AttendanceID = 3)

DELETE FROM Attendance

WHERE AttendanceID = 3;

* Other Queries

1. Retrieve Employees Managed by a Specific Manager:

-- Retrieve employees managed by a specific manager (e.g., ManagerID = 3)

SELECT \* FROM Employee WHERE ManagerID = 3;

1. Count the Number of Employees in Each Department:

-- Count the number of employees in each department

SELECT Department.Name, COUNT(Employee.EmployeeID) AS EmployeeCount

FROM Department

LEFT JOIN Employee ON Department.DepartmentID = Employee.DepartmentID

GROUP BY Department.Name;

1. Calculate the Average Salary in Each Department:

-- Calculate the average salary in each department

SELECT Department.Name, AVG(Employee.Salary) AS AverageSalary

FROM Department

LEFT JOIN Employee ON Department.DepartmentID = Employee.DepartmentID

GROUP BY Department.Name;

1. Find the Highest-Paid Employee:

-- Find the highest-paid employee

SELECT \* FROM Employee

WHERE Salary = (SELECT MAX(Salary) FROM Employee);

1. Calculate Total Salary Expenses:

-- Calculate the total salary expenses for the organization

SELECT SUM(SalaryAmount) AS TotalSalaryExpenses FROM Salary;