DATABASE DESIGN DOCUMNETATION

Domain: Flexible Work Arrangements for Employees

In modern workplaces, the concept of flexible work arrangements has gained significant importance. It involves providing employees with options such as remote work, flexible hours, and alternative work schedules. This approach aims to enhance employee satisfaction, work-life balance, and overall productivity. The system manages various aspects, including employee information, departmental structure, work schedules, and requests for flexible work.

Background Study and Requirement Analysis

A thorough background study was conducted to understand the requirements for managing flexible work arrangements. Key aspects include:

- **Remote Work:**

  Employees have the option to work from a location outside the traditional office. This may include working from home, co-working spaces, or other remote locations.

- **Flexible Hours:**

  Employees can customize their daily working hours, allowing them to start and end work at times that suit their individual preferences. This flexibility accommodates variations in peak productivity times and personal schedules.

- **Employee Management**:

  Understanding employee details, roles, and department affiliations.

- **Work Schedules**:

  Defining various types of work schedules and tracking schedules for each employee.

- **Requests Handling**:

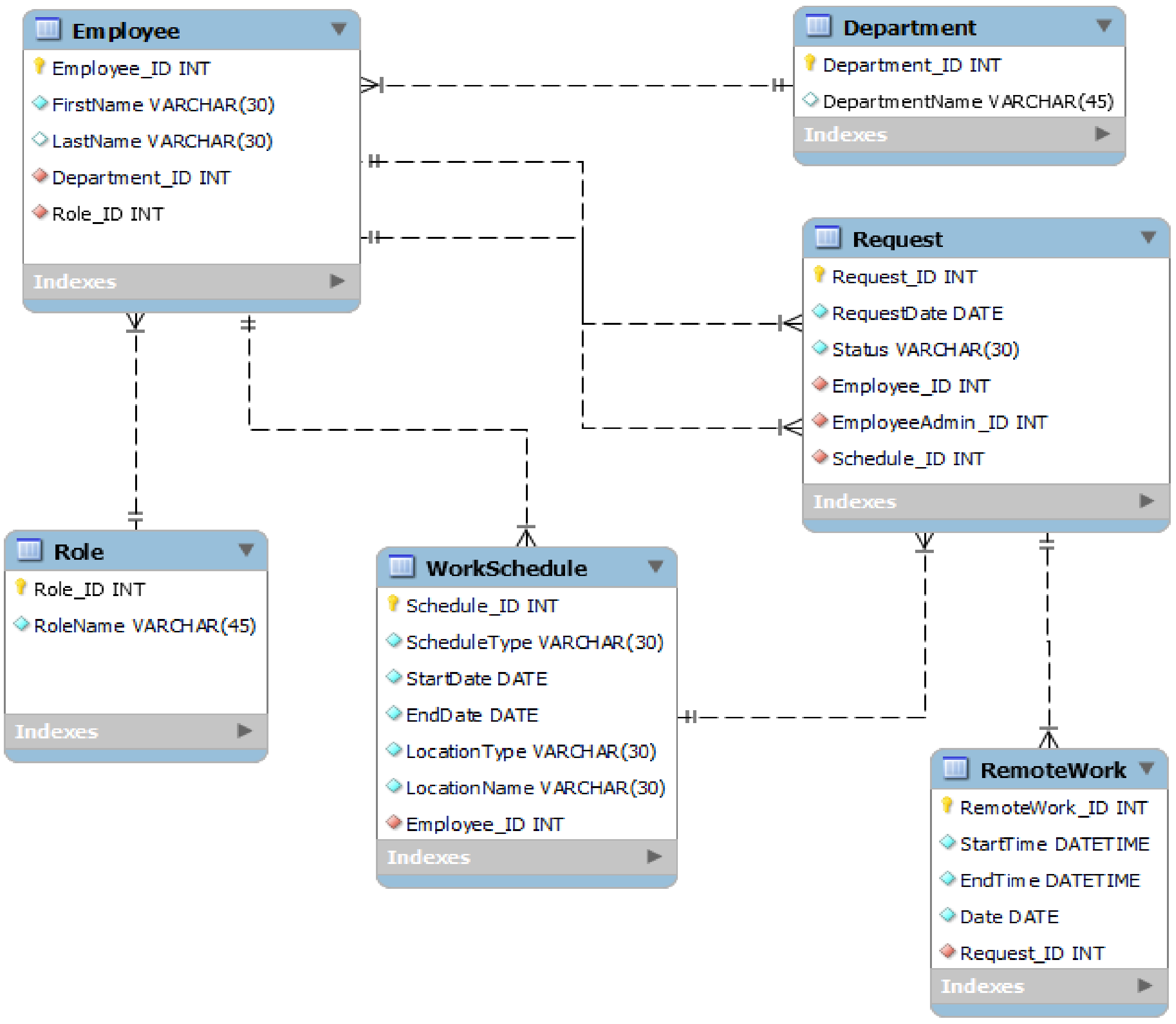
Managing requests from employees for flexible work arrangements.

- **Remote Work:**

  Recording details of remote work sessions initiated through approved requests.

Entity-Relationship Diagram (ERD)

The ERD below illustrates the relationships between tables:



Tables

- **Department**

  Attributes:

  `Department_ID` (Primary Key, INT)

  `DepartmentName` (VARCHAR)

- **Role**

  Attributes:

  `Role_ID` (Primary Key, ZEROFILL INT)

  `RoleName` (VARCHAR)

- **Employee**

  Attributes:

  `Employee_ID` (Primary Key, INT)

  `FirstName` (VARCHAR)

  `LastName` (VARCHAR)

  `Department_ID` (Foreign Key referencing `Department.Department_ID`)

  `Role_ID` (Foreign Key referencing `Role.Role_ID`)

- **WorkSchedule**

Attributes:

`Schedule_ID` (Primary Key, INT)

`ScheduleType` (VARCHAR)

`StartDate` (DATE)

`EndDate` (DATE)

`LocationType` (VARCHAR)

`LocationName` (VARCHAR)

`Employee_ID` (Foreign Key referencing `Employee.Employee_ID`)

- **Request**

Attributes:

`Request_ID` (Primary Key, INT)

`RequestDate` (DATE)

`Status` (VARCHAR)

`Employee_ID` (Foreign Key referencing `Employee.Employee_ID`)

`EmployeeAdmin_ID` (Foreign Key referencing `Employee.Employee_ID`)

`Schedule_ID` (Foreign Key referencing `WorkSchedule.Schedule_ID`)

- **RemoteWork**

Attributes:

`RemoteWork_ID` (Primary Key, INT)

`StartTime` (DATETIME)

`EndTime` (DATETIME)

`Date` (DATE)

`Request_ID` (Foreign Key referencing `Request.Request_ID`)

Normalization Process

1. Department Table:

The Department table stores information about different departments, such as Department_ID and DepartmentName. This table serves as a reference for departmental information, facilitating the categorization and organization of employees.

1NF:

- All attributes are atomic without repeating groups.
- Primary Key Department_ID uniquely identifies each department record.

2NF:

- No partial dependencies. All non-prime attributes depend on the entire primary key (Department_ID).

3NF:

- No transitive dependencies. All non-prime attributes depend only on the primary key.

2. Role Table:

The Role table, with the primary key Role_ID, contains details about different roles within the organization, including RoleName. It establishes a normalized structure to efficiently manage and assign roles to employees.

1NF:

- All attributes are atomic without repeating groups.
- Primary Key Role_ID uniquely identifies each role record.

2NF:

- No partial dependencies. All non-prime attributes depend on the entire primary key (Role_ID).

3NF:

- No transitive dependencies. All non-prime attributes depend only on the primary key.

3. Employee Table:

The Employee table, identified by the primary key Employee_ID, includes attributes like FirstName, LastName, Department_ID, and Role_ID. This table establishes foreign key relationships with the Department and Role tables.

1NF:

- All attributes are atomic without repeating groups.
- Primary Key Employee_ID uniquely identifies each employee record.

2NF:

- No partial dependencies. All non-prime attributes depend on the entire primary key (Employee_ID).

3NF:

- No transitive dependencies. All non-prime attributes depend only on the primary key.

4. WorkSchedule Table:

The WorkSchedule table, with the primary key Schedule_ID, captures information about employee work schedules, including ScheduleType, StartDate, EndDate, LocationType, LocationName, and Employee_ID.

1NF:

- All attributes are atomic without repeating groups.
- Primary Key Schedule_ID uniquely identifies each work schedule record.

2NF:

- No partial dependencies. All non-prime attributes depend on the entire primary key (Schedule_ID).

3NF:

- No transitive dependencies. All non-prime attributes depend only on the primary key.

5. Request Table:

The Request table, identified by the primary key Request_ID, manages requests. It includes attributes such as RequestDate, Status, Employee_ID, EmployeeAdmin_ID, and Schedule_ID.

1NF:

- All attributes are atomic without repeating groups.
- Primary Key Request_ID uniquely identifies each request record.

2NF:

- No partial dependencies. All non-prime attributes depend on the entire primary key (Request_ID).

3NF:

- No transitive dependencies. All non-prime attributes depend only on the primary key.

6. RemoteWork Table:

The RemoteWork table, with the primary key RemoteWork_ID, stores information about remote work instances, including StartTime, EndTime, Date, and Request_ID.

1NF:

- All attributes are atomic without repeating groups.
  Primary Key RemoteWork_ID uniquely identifies each remote work record.

2NF:

- No partial dependencies. All non-prime attributes depend on the entire primary key (RemoteWork_ID).

3NF:

- No transitive dependencies. All non-prime attributes depend only on the primary key.