`

# Introduction

This report presents the design and implementation of a web-based database application for an organization that plans to establish an online voting system to choose Employee of the Month, according to the case study given the coursework. We are required to make a system where Employees are not allowed to vote themselves also one employee can work on one department only but allowed to vote an employee from any department. There is different role of employees in different department, and everyone is eligible to vote. In Organization, each department have their own manager who are also an employee of that organization and organization keep the record of employee job history if any employee changes his/her department and job. The report covers the requirements of the coursework, including Normalization for given two figures, E-R Model, Data Dictionary, Generation of Database, and Implementation of Web-based Database Application. The project is done with fulfilling proper requirement of the coursework like developing the system prototype using Oracle SQL Developer, Data Modeler, visual studio and ASP.NET with C#.

* **Oracle SQL Developer**



Figure 1: Oracle SQL Developer

(Oracle, 2023) Oracle SQL Developer is a free, all-in-one tool for designing and maintaining Oracle databases in both traditional and cloud environments. This software automates the entire process of developing PL/SQL applications. It includes a workspace for running queries and scripts, a database management console, a reporting interface, a full data modeling solution, and a migration platform for migrating third-party databases to Oracle.

* **Data Modeler**



Figure 2: Data Modeler (oracle, 2023)

(oracle, 2023) Oracle SQL Developer Data Modeler is a piece of software that speeds up and simplifies data modeling. It is a graphical tool for creating, editing, and viewing various types of models, such as logical, relational, physical, multi-dimensional, and data type models. The program facilitates collaborative development and provides forward and reverse engineering capabilities by utilizing integrated source code control. It is also meant to be used in both traditional and cloud environments. With Oracle SQL Developer Data Modeler, users may expedite data modeling procedures and increase productivity.

* **Visual Studio**



Figure 3: Visual Studio

(Anshul\_Aggarwal, 2023) Microsoft's Visual Studio is a software development tool that includes an Integrated Development Environment (IDE) for creating graphical user interface (GUI), console, web, mobile, cloud, and web services applications. This IDE enables developers to write both managed and native code, and it supports many Microsoft software development tool platforms such as Windows Store, Microsoft Silverlight, and Windows API. In contrast to other language-specific IDEs, Visual Studio supports a wide range of programming languages, including C#, C++, Visual Basic (VB), Python, and JavaScript, among others.

# Textual Analysis

## 2.1. Textual Analysis 1

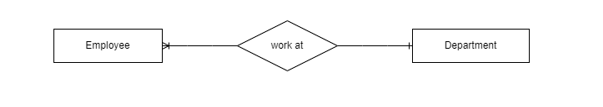


Figure 1 Textual Analysis of Employee and Department

**Description:** One employee is assigned to one department. One department can have at least one employee or many employees.

## 2.2. Textual Analysis 2

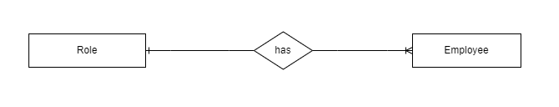


Figure 2 Textual Analysis of Role and Employee

**Description:** Many employees is assigned to at least a role (for instance: “Manager”, “Supervisor”, “Staff”). A role can have at least one employee of many employees.

## 2.3. Textual Analysis 3

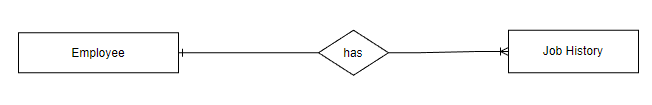


Figure 3 Textual Analysis of Employee and Job History

**Description:** The relationship between a job history and an employee is represented using a “one-to-many” relationship, where one employee can have multiple job histories.

## 2.4. Textual Analysis 4

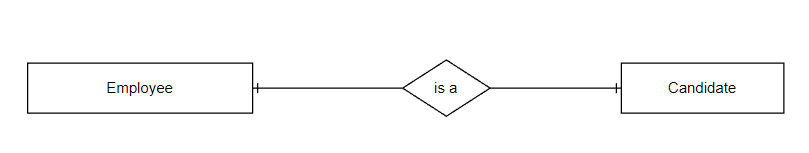


Figure 4 Textual Analysis of Employee and Candidate

**Description:** The relationship between employee and candidate is represented using a “one-to-one” relationship, where one employee can be a candidate, and a candidate must be among employee.

## 2.5. Textual Analysis 5

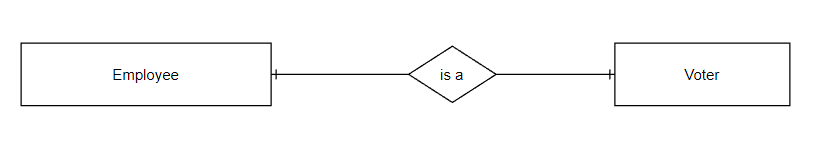


Figure 5 Textual Analysis of Employee and Voter

**Description:** The relationship between employee and voter is represented using “one-to-one” relationship, where one employee can be a voter, and a voter must be among employees.

## 2.6. Textual Analysis 6

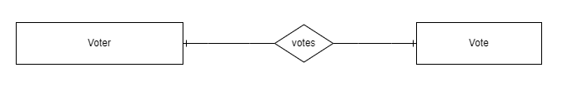


Figure 6 Textual Analysis of Voter and Vote

**Description:** The relationship between voter and vote is represented using “one-to-one” relationship where one voter can cast only one votes, and one vote can be cast by one voter.

## 2.7. Textual Analysis 7

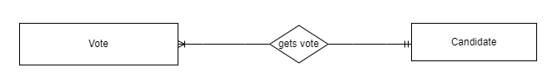


Figure 7 Textual Analysis of Vote and Candidate

**Description:** The relationship between candidate and vote is represented using “one-to-many” relationship where one candidate can receive multiple votes, and many votes can be casted for one candidate.

## 2.8. Textual Analysis 8

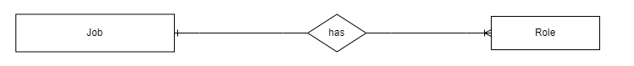


Figure 8 Textual Analysis of Job and Role

**Description:** The job and role entities are represented using one-to-many relationship, where a job can have multiple roles associated with it, but a role can only be associated with one job.

# ERD from Case Study

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities (visual-paradigm, 2023).

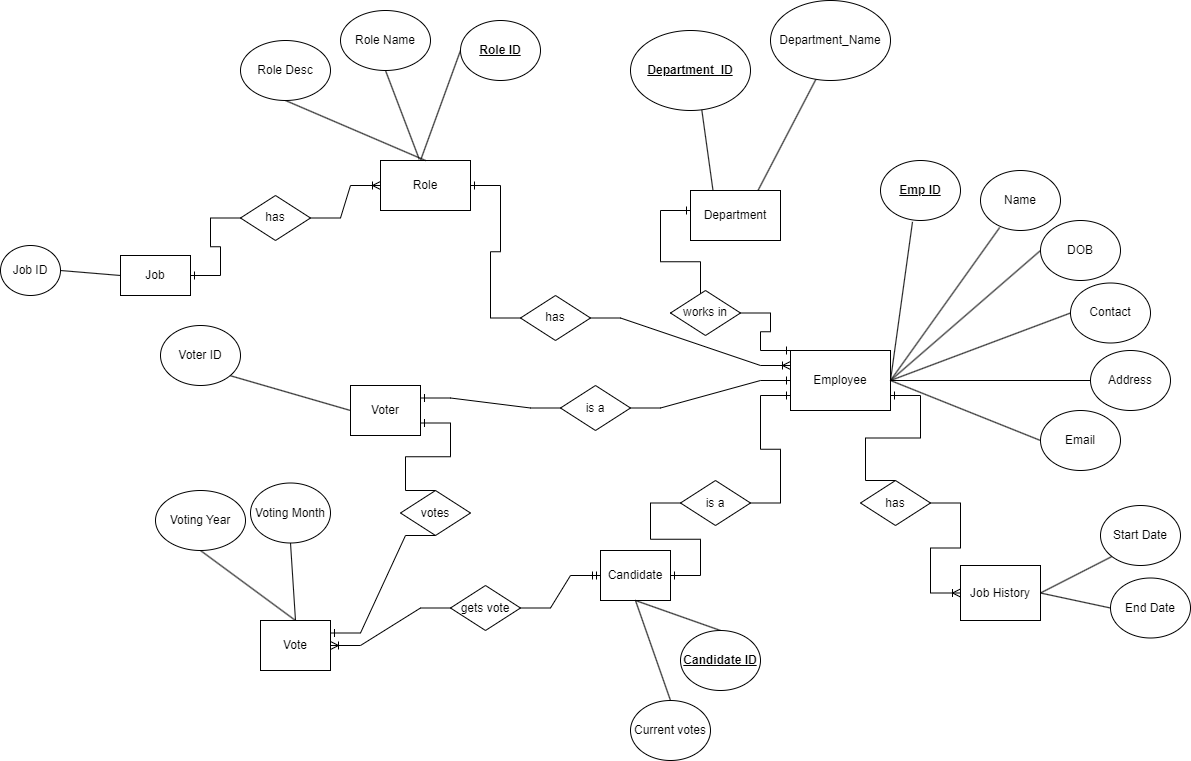


Figure 9 ERD from Case Study

# Normalization

Normalization is a method used to ensure that the data stored in a database satisfies two primary conditions:

1. There is no repetition of data, and each data item is stored only once.
2. Data relationships are coherent, and all connected data items are grouped together.

Normalization is critical for several reasons, primarily because it helps databases take up minimal disk space, leading to improved performance. It is also referred to as data normalization.

The three main types of normalization are listed below:

1. **First Normal Form(1NF)**

There are specific regulations that tables must comply with to be considered in the First Normal Form (1NF):

* Each cell within the table must consist of only one (atomic) value.
* All columns in the table must have a unique name.
* Every value in a column must belong to the same domain.

1. **Second Normal Form(2NF)**

For a table to be in the Second Normal Form (2NF), it must already comply with the rules of the First Normal Form (1NF), and it must not have any partial dependencies. In other words, all non-prime attributes must rely entirely on the table's primary key.

1. **Third Normal Form(3NF)**

To meet the requirements of the Third Normal Form (3NF), a table must fulfil the prerequisites of the Second Normal Form (2NF) and must not have any transitive functional dependencies with regards to the primary key. Put differently, all non-prime attributes should depend solely on the primary key, without relying on other non-prime attributes.

## 4.1 Normalization of fig 1(Employee Details)

**UNF**

**Employee** (employeeID, employeeName, employeeDOB, employeeContact, {employeEmail } ,{address}, departmentID,department)

**Applying 1NF – Remove Repeating Group**

For getting the tables in 1NF form we need to remove the repeating group which exists in the above UNF Employee table. After removing the repeating groups, we get,

Final 1NF

**Employee-1** (employeeID, employeeName, employeeDOB, employeeContact, departmentID, department)

**Address-1 (**addressID, address, employeeID\*)

**Email-1** (employeeID\*,employeeEmail)

**Applying 2NF- Remove Partial Functional Dependency**

A partial dependency would occur whenever a non-prime attribute depends functionally on a part of the given candidate key. Or if there is composite primary key is present.

**Checking** the PFD in Employee-1 Table,

It is already in 2NF as it has only a single attribute as Unique Identifier and it exclude a composite key likewise it does not have Partial Functional Dependency.

**Checking** the PFD in Email-1 Table,

It is already in 2NF as it has only a single attribute as Unique Identifier and it exclude a composite key likewise it does not have Partial Functional Dependency.

**Checking** the PFD in Address-1 Table,

Address-1(address, addressID, employeeID\*)

**There exist** PFD in Address-1 Table as,

addressID → address

addressID, employeeId→X

Therefore, Tables in 2NF,

**Employee-**2 (employeeID, employeeName, employeeDOB, employeeContact, departmentID, department)

**Address-2** (addressID, address)

**Employee\_Address-2** (addressID\*, employeeID\*)

**Email-2** (employeeID\*, employeeEmail)

**Applying 3NF – RemovingTransitive Dependency**

**Checking** transitive dependency in Employee-2 table:

empId → deptId → department

**Checking** transitive dependency in Email-2 table:

Email-2 table is already in 3NF because there does not exist a transitive dependency.

**Checking** transitive dependency in Address-2 table:

Address-2 table is already in 3NF because there does not exist a transitive dependency.

**Checking** transitive dependency in Employee\_Address-2 table:

Employee\_Address table is already in 3NF because there does not exist a transitive dependency.

**Therefore, Final 3NF**

**Employee-3** (employeeID,employeeName,dateOfBirth,contact,departmentID\*)

**Email-3** (employeeEmail, empIoyeeID\*)

**Address-3** (addressID, address)

**Employee\_Address-3** (employeeID\*, addressID\*)

**Department-3** (departmentID, departmentName)

## 4.2 Normalization of fig 2. (Voting Records)

**Applying UNF**

**Voting\_Record** (voterID, voterName, {votingYear {votingMonth, candidateID, candidateName, candidateDepartment}})

**Applying 1NF- Remove Repeating Group**

For getting the tables in 1NF form we need to remove the repeating group which exists in the above UNF Employee table. After removing the repeating groups, we get,

**Voting\_Record\_1** (voterID, voterName)

**Voting\_Year\_1** ( votingYear, voterID\*)

**Candidate\_1** (candidateID, candidateName, candidate, votingMonth, VoterID\* ,voting\_year\*)

**Therefore, Final 1NF**

**Voting\_Record\_1** (Voter\_id, Voter\_name)

**Voting\_Year\_1** (Voting\_year ,Voter\_id\*)

**Candidate\_1** (Candidate\_id, candidate\_name, candidate\_department, voting\_month, voter\_id\*, voting\_year\*)

**Applying 2NF – Remove Partial Functional Dependency**

**Voting\_Record\_1** (Voter\_id, Voter\_name)

* **This Relation is already in 2NF as it has only a single attribute as Unique Identifier and it excludes composite key, therefore, it does not have Partial Functional Dependency**

**Candidate\_1** (Candidate\_id, candidate\_name, candidate\_department, voting\_month, voter\_id\*, voting\_year\*)

* **This relation may or may not have Partial Functional Dependency. So, need to check. Put the Full Primary Key as the Determinant and all the parts of the primary key as the Determinant and check the remaining attributes’ dependency.**

**Voting\_month à**

* **This has no Functional Dependent Attribute therefore do not have Partial Functional Dependency. No need to create a new entity.**

**voting\_year à**

* **This has no Functional Dependent Attribute therefore do not have Partial Functional Dependency. No need to create a new entity.**

**voter\_id🡪**

* **This has no Functional Dependent Attribute therefore do not have Partial Functional Dependency. No need to create a new entity.**

**Voting\_id, voting\_month 🡪**

* **This has no Functional Dependent Attribute therefore do not have Partial Functional Dependency. No need to create a new entity.**

**Voting\_month, voting\_year 🡪**

* **This has no Functional Dependent Attribute therefore do not have Partial Functional Dependency. No need to create a new entity.**

**Voter\_id,voting\_year 🡪**

* **This has no Functional Dependent Attribute therefore do not have Partial Functional Dependency. No need to create a new entity.**

**Voting\_month,voter\_id,voting\_year 🡪 candidate\_id, candidate\_name, candidate\_department**

* **This has Functional Dependent Attribute Where Partial Functional Dependency exists. Create this as a new entity with a new name and the Determinant will be a unique Identifier.**

**Therefore, Final 2NF**

**Voting\_Record\_2** (Voter\_id, Voter\_name)

**Voting\_Year\_2** (Voting\_year ,Voter\_id\*)

**Voting\_Record\_Candidate-2** (voter\_id\*,voting\_year\*, voting\_month, candidate\_id, candidate\_name, candidate\_department)

**Applying 3NF- Remove Transitive Dependency**

**Voting\_Record\_2** (Voter\_id, Voter\_name)

* **This relation is already in 3NF as it has only one key attribute**

**Voting\_Year\_2** (Voting\_year ,Voter\_id\*)

* **This relation is already on 3NF as it does not have any key attribute.**

**Voting\_Record\_Candidate-2** (voter\_id\*,voting\_year\*, voting\_month, candidate\_id, candidate\_name, candidate\_department)

* **This relation may have Transitive Dependency because in the voting\_record\_candidate-2 table there are more than 1 non-key element,so it needed to be checked to remove transitive dependency**

**candidate\_id 🡪 candidate\_name 🡪 candidate\_department**

**Therefore, Final 3NF**

**Voting\_Record\_3** (Voter\_id, Voter\_name)

**Voting\_Year\_3** (Voting\_year ,Voter\_id\*)

**Candidate\_3 (**Candidate\_id**,** candidate\_name, candidate\_department**)**

**Voting\_Record\_Candidate-3** (candidate\_id\*,voter\_id\*,voting\_year\*, voting\_month)

# Data Dictionary

1. Table: **Address**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| addressID | Varchar | 30 | Primary Key |  |  | To store a unique id for addresses. | Ad12 |
| city | Varchar | 30 |  |  |  | To store the city name. | Kathmandu |
| state | Varchar | 20 |  |  |  | To store the state name. | Lumbini |
| district | Varchar | 30 |  |  |  | To store the district name. | Kathmandu |
| zipCode | Varchar | 30 | unique |  |  | To store the zipCode name. | 44000 |

1. Table: **Department**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| departmentID | Varchar | 30 | Primary Key |  |  | To store a unique id for every department. | D89234 |
| departmentName | Varchar | 40 |  |  |  | To store the name of the department. | HR |
| departmentEmail | Varchar | 50 |  |  |  | To store the email address of the department. | Hr.organization@gmail.com |
| departmentPhoneNumber | Varchar | 15 |  |  |  | To store phone number of the department. | +9772348762234 |

1. Table: **Department\_Manager**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| departmentID | Varchar | 30 | Foreign Key | Department | departmentID | To identify the department of the manager | W000 |
| employeeID | Varchar | 25 | Foreign Key | Employee | employeeID | To identify the manager of the department and other employees of that department. | Mg2983 |
| startDate | Date |  | Mandatory |  |  | To store the work, start date of the manager. | 12-march-2023 |
| endDate | Date |  |  |  |  | To store the work end date of the manager | NULL, 12-march-2025 |

1. Table: **Employee**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| employeeID | Varchar | 25 | Primary Key |  |  | To uniquely identify Each employee. | Em2398 |
| employeeName | Varchar | 30 | Not null |  |  | To store the name of employee. | Anish |
| employeeDOB | Date |  |  |  |  | To store date of birth of the employee. | 12-march-2023 |
| employeeContact | Varchar | 15 | Unique |  |  | To store the contact number of employee. | +977988133452 |
| departmentID | Varchar | 30 | Foreign key | Department | departmentID | To store the department id of the respective employee linking them to their respective department. | Dp32489 |
| roleID | Varchar | 20 | Foreign Key | Role | roleID | To store the role of respective employee corresponding to the record of Role table. | R23888 |

1. Table: **Employee\_Address**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| addressID | Varchar | 30 | Foreign Key | Address | addressID | To store the unique address id for employee which corresponds to the record of Address table. | Ad2342 |
| employeeID | Varchar | 25 | Foreign Key | Employee | employeeID | To store the unique employee id for employee which corresponds to the record of employee table which links the employee table and address table. | Em3248 |

1. Table: **Employee\_Email**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| emailID | Varchar | 30 | Primary key |  |  | To store unique id for each email of the employees. | anisahryal058@gmail.com |
| employeeID | Varchar | 25 | Foreign Key | Employee | employeeID | To store employeeID of the employee together with the email details which gives the email of the respective employee linking employee and employee\_email table. | Em23984 |

1. Table: **Job Table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| jobID | Varchar | 20 | Primary key |  |  | To store the unique id for the jobs in the organization. | Jb9832 |
| jobName | Varchar | 20 |  |  |  | To store the name of the job that are present in the organization. | Admin Asst |
| minimumSalary | Integer |  |  |  |  | It stores the minimum salary for the respective job in the organization. | 150000 |
| maximumSalary | Integer |  |  |  |  | It stores the maximum salary for the respective job in the organization | 70000 |

1. Table: **Job History**

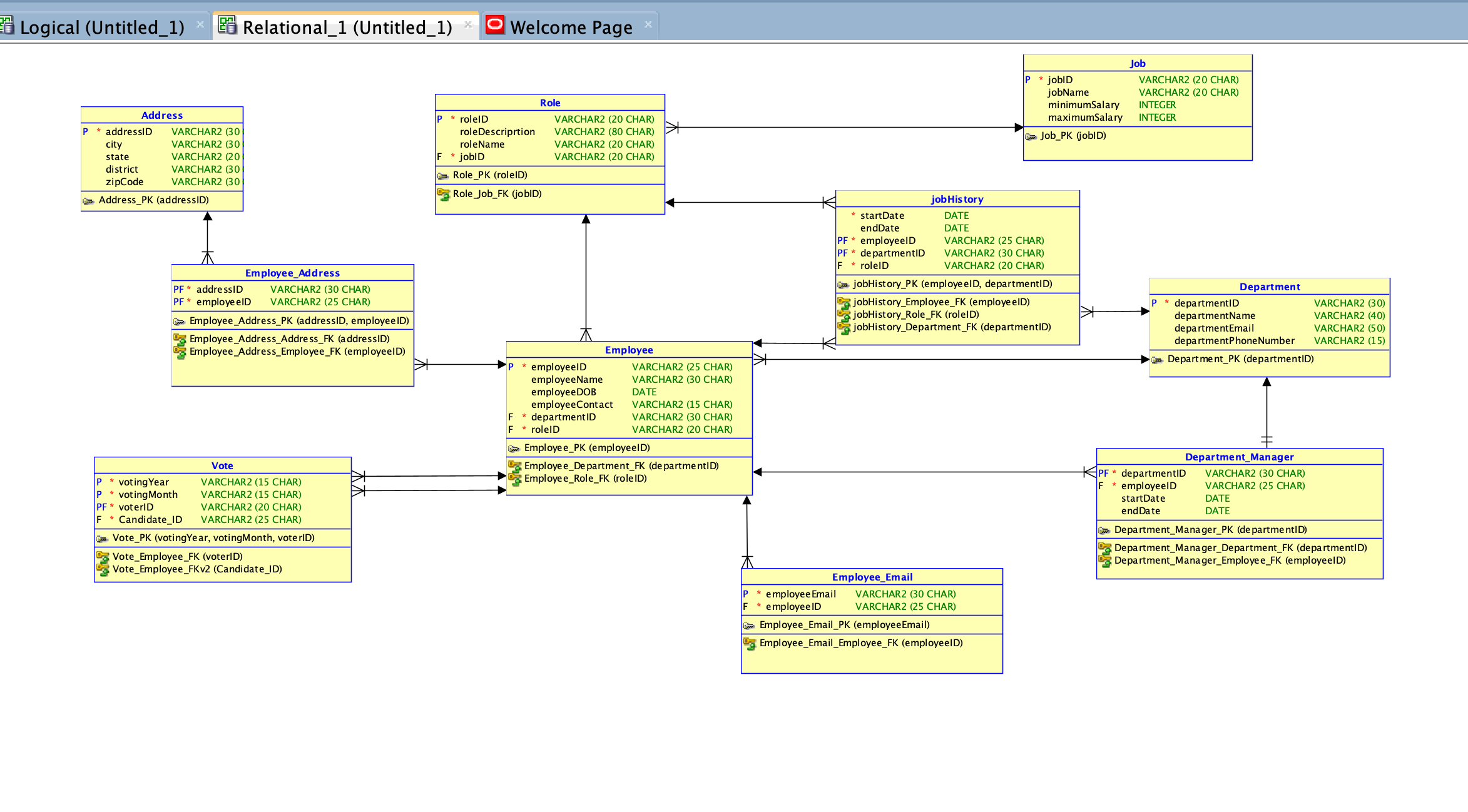
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| startDate | Date |  |  |  |  | To store the date where an employee started their job. | 12-march-2023 |
| endDate | Date |  |  |  |  | To store the date when an employee ended their job. | NULL, 12-march-2023 |
| employeeID | Integer | 25 | Foreign key | Employee | employeeID | To store the unique id of employee which references to the records of employee table. | Em32498 |
| departmentID | Integer | 30 | Foreign key | Department | departmentID | To store the unique id of the department which references to the records of the department table. | Dp3422 |
| roleID |  | 20 | Foreign key | Role | roleID | To store the unique id of the role which references to the records of the role table. | R42389 |

1. Table: **Role**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| roleID | Varchar | 20 | Primary key |  |  | To store the unique id for the role of the employee of the organization. | R3422 |
| roleName | Varchar | 20 |  |  |  | To store the name of the respective role. | UI/UX designer |
| roleDescription | Varchar | 80 |  |  |  | To provide additional information about the role the employee is playing. | Design the ux and ui of the product to make it as user friendly as possible. |
| jobID | Varchar | 20 | Foreign key | Job | jobID | To store the unique id of the job which references to the records of the job table. | Jb23897 |

1. Table: **Vote**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| Voting\_year | Varchar | 20 | Primary key |  |  | It stores the year in which the vote was casted. | 2023 |
| Voting\_month | Varchar | 20 | Primary key |  |  | It stores the month in which the vote was casted. | march |
| Voter\_id | Varchar | 10 | Foreign key | Employee | employeeID | To store the voters id of the employee which help keep track of the employees who casted the vote and who didn’t. | E923845 |
| Candidate\_id | Varchar | 30 | Foreign key | Employee | employeeID | To store the vote casted by the employees to the respective candidate which represents the employee of the EmployeeID. | E29835 |



# Conclusion

After finishing the project, it is safe to say that the intended result, as described by the coursework requirements, was accomplished. I was able to expand my understanding of the ideas related to SQL developer, SQL queries, data modeling, and data handling by applying the theoretical concepts covered in class to actual, real-world problems during the project.

The major goal of the project was to create user-friendly software that would allow a company to elect its employee of the month via an online voting process. In addition to learning how to build a database architecture and perform SQL queries to obtain and change data, I also gained knowledge of how to implement user interfaces that offer end users a seamless user experience. I succeeded in doing the task on schedule with proper time and risk management.

In conclusion, the completion of this project provided a significant learning opportunity, allowing me to apply theoretical concepts to real-world situations, develop practical skills, and gain project management experience. This experience will be beneficial in my future academic and professional pursuits.