# Contact Management System

**Database Management Systems**
**MSCS 542L**

**Marist Database Administrators (DBA's)**

Marist College
School of Computer Science and Mathematics

Submitted To:
Dr. Reza Sadeghi

Fall 2022

# Project Report of Contact Management System

## Team Name
Marist DBA’s

## Team Members

1. Noah Carbo - noah.carbo1@gmail.com - (845) 489-8296 - (Team Head)
2. Theresa Gundel - theresa.gundel1@marist.edu - (518) 526-8737 - (Team Member)
3. Ryan Weidling - ryan.weidling1@marist.edu - (215) 630-7789 - (Team Member)
4. Dale Doster - dale.doster1@marist.edu - (845) 518-2405 - (Team Member)
5. Sai sumanth Gurrala - saisumanthreddy.gurrala1@marist.edu - (Team Member)
6. Thomas Diaz-Piedra - thomas.diaz-piedra1@marist.edu - (201) 824-3695 - (Team Member)
7. Banu Pitta Reddy - banuprakash.pittareddy1@marist.edu - (845) 214-3986 - (Team Member)

## Description of Team Members
1. Noah Carbo
    a. My name is Noah Carbo, I am working on my master’s in Software Development at Marist College. I wanted to work with my group based on their proximity to where I sat on the first day of class. Everyone on the team seems very competent in their fields and we have agreed on communicating through Whatsapp while working on this project. I am looking forward to working with everyone this semester.
2. Theresa Gundel
    a. I graduated from Binghamton University in 2021 with a Bachelor’s in Computer Science. Now I am pursuing a Master’s in Computer Science at Marist College. I did not know anyone in the class beforehand so I chose to work with this group because we were sitting near each other in class. We chose Noah as Team Head because he volunteered first and seemed like a good fit.
3. Ryan Weidling
    a. My name is Ryan and I am working on my second degree from Marist. I graduated in 2018 with a Bachelor’s in Communications and am now pursuing a Master’s in Computer Science to pursue a career in software engineering. I wanted to work with the current group members because of two reasons. One, they were in close proximity to me, and two, each person at some point in the first two weeks had demonstrated qualities that would make them great teammates. Whether that was showing

friendliness by introducing themselves when we sat down, or sharing their background and proving their technical prowess. We chose Noah as Team lead because when the project was announced he was already brainstorming ideas for what our group should be working on.

4. Dale Doster
    a. In 2020, I graduated with a degree in Computer Science from Marist College. My focus in undergrad was on Software Development. After having worked as a Software Engineer for the past couple years, I decided to pursue a Master's degree, again with a focus on Software Development. I decided to work with my team because they were sitting near me and were easily approachable. We decided to select Noah as the Team Head because he volunteered and has good communication skills.
5. Sai sumanth Gurrala
    a. I am Sai Sumanth, I graduated with a bachelors in computer science from Bharath University, India. I am currently working with my group members for the Database management project, because we were sitting together in class. And it made more sense once we got connected through a whatsapp group and shared our interests and work. I am confident that my team members are very competitive and together we can bring the most out of ourselves for the project of contact management system. We chose Noah as our team head, because we felt he was more capable of leading our team towards our goal.
6. Thomas Diaz-Piedra
    a. My name is Thomas. I graduated from Marist with a bachelors in computer science and minors in information systems and information technology last year. I wanted to work with my group mates because we all sat near each other on the first couple days of classes and we all seemed to have the same outlook when it came to this project. We chose Noah as the team head/leader as he volunteered to do it and seemed more than capable of doing the job. Really looking forward to working with this team this semester.
7. Banu Pitta Reddy
    a. My name is Banu Prakash, I graduated with a bachelors in Mechanical engineering from Jawaharlal Nehru Technological University, India. I have worked as a Software Quality Engineer for almost 9 years and decided to pursue my masters degree with focus on Cloud Computing. I met only a few mates in the class and found these people enthusiastic about the work. We spent some time sharing our interests and work, where I got to know more about my team members. Everyone in the team looks to be goal oriented and working towards a common objective. We chose Noah

to head the team as he volunteered to do it and we trust his efficiency with the level of confidence that he demonstrates. I am very excited to join the team and looking forward to a great collaboration in completing this project.

## Table of Contents

Table of Figures …………………………………………………………………………….. 5
Project Objective …………………………………………………………………………..… 7
Review the Related Work ……………………………………………………………….. 8
The Merits of our Project ………………………………………………………………..…. 9
GitHub Repository Address ……………………………………………………………… 10
Entity Relationship Model (ER Model) ………………………………………………….. 11
Enhanced Entity Relationship Model (EER Model) ……………………………………… 13
Database Development ……………………………………………………………….. 15
References …………………………………………………………………………………… 21

## Table of Figures

Figure 1: ER Model

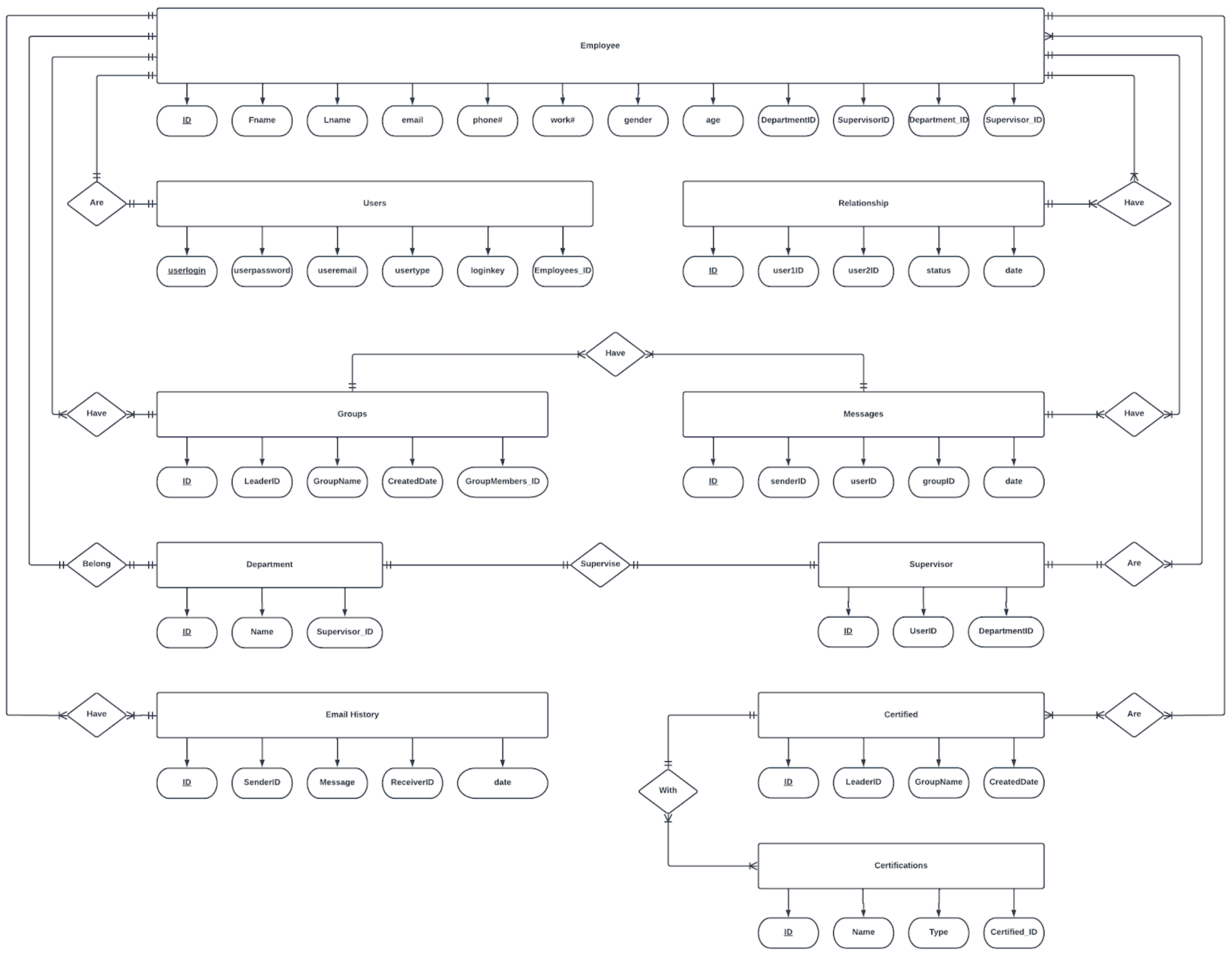
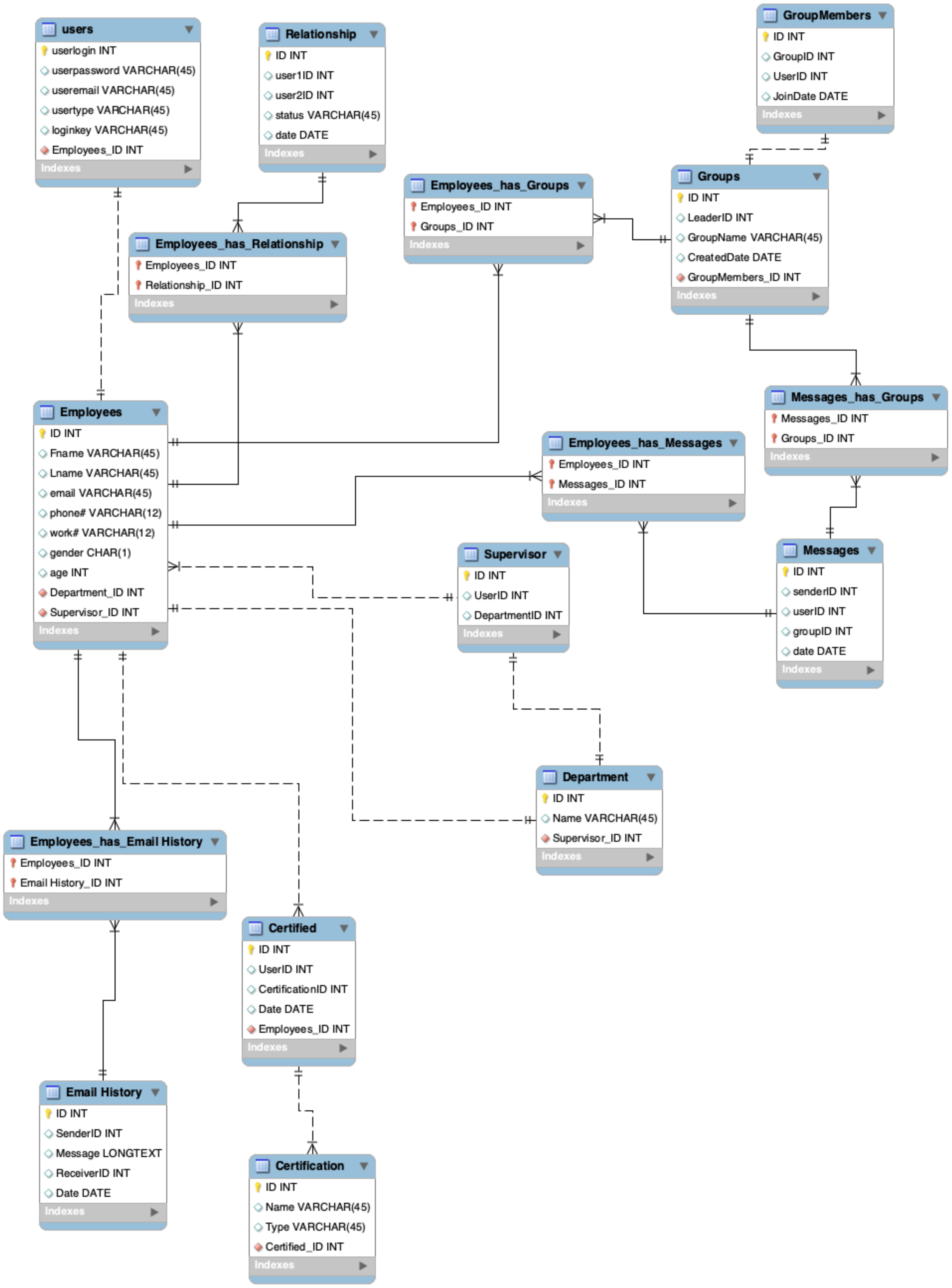


Figure 2: EER Model



## Project Objective

The Contact Management System (CMS) was conceived to allow for storage of any and all information that a user wants to store. These could include phone numbers, home addresses, email addresses and other information which are stored securely and distinctly from other user information. This system at a minimum supports the following functions:

1. Allows for admin access with a secure credentials and include the following abilities
    a. To add/edit additional admins
    b. Add and delete user profiles
    c. Edit and manage user information
    d. Allows for resetting usernames/passwords in the event the user locks themselves out of their accounts
2. Allows for users to customize their stored information with the following capabilities
    a. Add/Delete multiple contact information
    b. Allows for editing of saved contact information
    c. Contacts can be searched based on their specific details
3. The CMS has a Graphical User Interface (GUI) which allows for interaction between the user and the database
    a. The GUI prevents duplicate contacts to be added
    b. It is user friendly including a welcome screen, provides reports in a tabular form and allows the user to exit the program.
4. Sensitive user information is ciphered so that in the unfortunate event of a hack of our user information the data that is collected would be indecipherable.

## Review the Related Work

One example of an existing Contact Management System (CMS) is Streak. This CMS is unique because it can integrate with Google products [1]. This can be good for people who already use Google accounts, but for people who don’t this is not that useful. A positive aspect of Streak is that it can integrate with Gmail to show the contact information, such as the company, of the person you’re emailing. It can also show a timeline of all communications with a contact including emails, call logs, and files shared. Streak also has a good system for searching, sorting, filtering, and grouping contact data [1]. The main negative aspect of Streak is that it’s focused on Google products so if you don’t use a Google account or Google Chrome then this CMS is not ideal [4].

Another example of a CMS is Nimble. Nimble has many positive aspects. For example, you can import contacts into Nimble from your Gmail and Outlook accounts as well as from an uploaded CSV file. Another cool feature is that Nimble can scan email signatures and use that information to keep your contacts up to date. A negative aspect of Nimble is that you’re limited to 25,000 contacts unless you want to pay to have more. Also, Nimble attempted to include email access in their tool but it lacks a lot of common functions like accessing folders [5].

A third example of a real-world CMS is Keap. A good feature of Keap is their user-friendly platform. They also have thousands of integration options such as Gmail, Outlook, and Quickbooks [2]. Based on these integrations, Keap automatically keeps all your contact data up to date [1]. A flaw that Keap has is it does not allow users to make calls from their platform. Users can make calls separately and log them after, but Keap does not automatically track calls [3].

## The Merits of our Project

Our contact management system will have a number of key capabilities provided for the end user that will entice them to utilize our CMS over a competitor. Our CMS will allow for both admin and standard user login, with admin access allowing for full control over additional admin processes, adding/deleting user profiles, editing/managing user information, and allowing for the resetting of usernames/passwords in the event the user locks themselves out of their accounts. Additionally, users will be able to customize their stored information with the following capabilities: add/delete multiple pieces of contact data, edit saved contact information, and search within the CMS. Our CMS will have a GUI which will allow for users to interact with the database. One of the main draws to our CMS will be the idea that the GUI will help prevent duplicate contacts from being added. On top of a functional GUI, our CMS will cipher sensitive user information so that in the unfortunate event of a hack, our users' information will be indecipherable and safe from bad actors.

## GitHub Repository Address

https://github.com/NoahCarboMarist/MSCS542_ContactManagementSystem_ContactManagers

## Entity Relationship Model (ER Model)

Our CMS is intended for a business to use. Based on this design decision, the entities are focused on workplace things, such as employees, groups (like teams), supervisors, and departments. We chose attributes that would typically describe each entity. For example, the Employee entity has the following attributes: first name, last name, email, phone number, etc. Some of these attributes are foreign keys like Employees have Department_ID and Supervisor_ID. The design mainly centers around employees' relationships to things such as logging into users, belonging to departments, and being supervisors. Our ER Model can be seen in Figure 1.

Descriptions:

The **Employee** entity describes a person employed at a company using our CMS. This person has a first name, last name, email address, phone number, work phone number, gender, and age. They also have an integer ID which serves as the primary key for this entity. There are foreign keys relating the Employee entity to its corresponding Department and Supervisor.

An Employee has a many to one relationship, "belong", with the Department entity. A **Department** has an integer primary key called ID. It has a foreign key corresponding to the ID of its Supervisor. It also has a name.

A Supervisor entity has a one to one relationship, "supervise", with the Department entity. It also has a one to one relationship, "are", with the Employee entity. The **Supervisor** entity has an integer ID as its primary key. It also has foreign keys corresponding to the Supervisor's Employee record and the Department they supervise.

An Employee entity has a one to many relationship, "are", with the **Certified** entity. This entity has an integer ID as the primary key as well as foreign keys corresponding to the Employee who received the certification and which Certification was received. There is also the date the employee was certified.

The Certified entity has a one to one relationship, "are", with the **Certification** entity. This entity has an integer ID as the primary key. It also has a name and type.

The Employee has a one to many relationship, "have", with the Relationship entity. The **Relationship** entity has an integer ID as its primary key. It also has two IDs for each employee in the relationship, a status, and a date.

The Employee entity has a one to one relationship, "login", with the User entity. The **User** entity has an integer as its primary key. It also has a password, email, type, and key. There is a foreign key to the ID in the corresponding Employee record.

The Employee entity has a one to many relationship, "send", with the **Message** entity. The Message entity has a primary key which is an integer called ID. It also has the sender ID, receiver ID, and group ID, as well as the date it was sent.

The Group entity has a one to many relationship, "has", with the Message entity. It also has a many to one relationship, "join", with the Employee entity. The **Group** entity has an integer ID for its primary key. It also has the ID of the leader of the group, the group name, and the date it was created. It also has a foreign key to the GroupMembers entity.

The **GroupMembers** entity has a many to one relationship, "contains", with the Group entity. The GroupMembers entity has a primary key which is an integer ID. It also has the group ID, user ID, and the date the user joined.

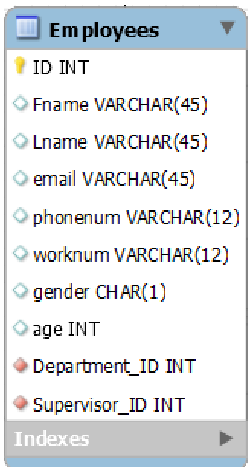
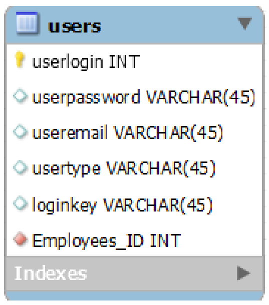
The Employee entity has a one to many relationship, "have", with the **Email History** entity. This entity has the IDs of the sender and receiver, as well as the date it was sent and the body of the email. The Email History has a unique integer ID as its primary key.

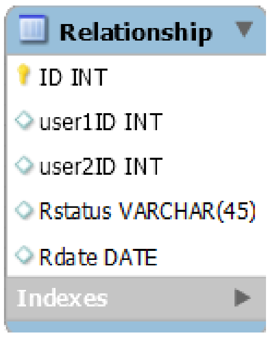
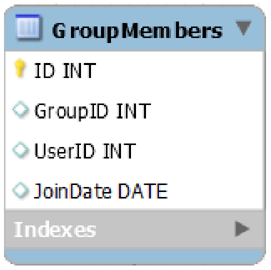
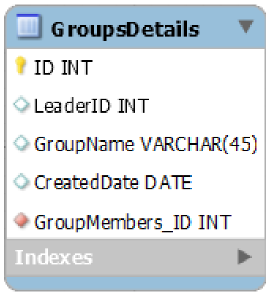
## Enhanced Entity Relationship Model (EER Model)

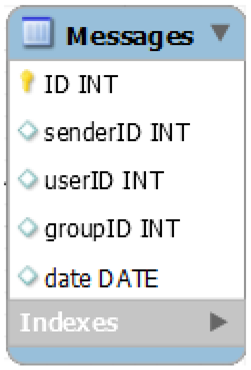
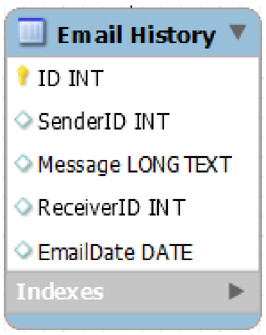
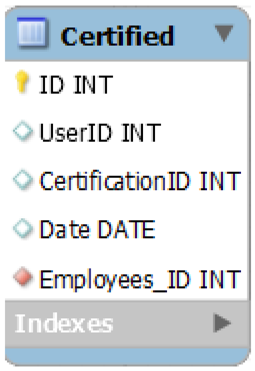
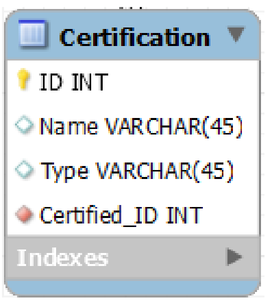
Keys and relationships play an integral part in relational databases. Keys are attributes (or a set of attributes) that help to identify a row uniquely in a table. Keys are used when you want to show that there is a relationship between different columns and tables of a relational database. There are several different types of keys: Primary Key, Candidate Key, Super Key, Foreign Key, Alternate Key, Composite Key, and Artificial Key. [6] A primary key contains values in a column that uniquely identify rows of data in a table. "A foreign key (FK) is a column or combination of columns that is used to establish and enforce a link between the data in two tables to control the data that can be stored in the foreign key table." [7] "A relationship, in the context of databases, is a situation that exists between two relational database tables when one table has a foreign key that references the primary key of the other table. Relationships allow relational databases to split and store data in different tables, while linking disparate data items." [8]

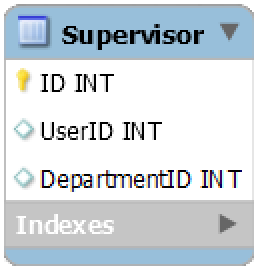
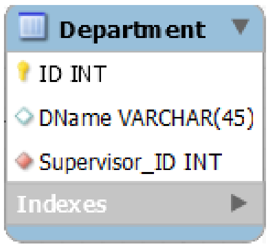
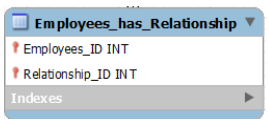
Our team was able to make use of numerous keys and relationships while designing our CMS, as seen in our EER Model in Figure 2. The employee entity has an integer ID which serves as its primary key. The employee entity has corresponding foreign keys for its relationships with Department and Supervisor. The Employee has a many to one relationship with the Department entity. The Department entity has an integer primary key called ID. It has a foreign key corresponding to the ID of its Supervisor. The Supervisor entity has a one to one relationship with the Department entity. It also has a one to one relationship, "are", with the Employee entity. The Supervisor entity has an integer ID as its primary key. It also has foreign keys corresponding to the Supervisor's Employee record and the Department they supervise. An Employee entity has a one to many relationship with the Certified entity. The Certified entity has an integer ID as its primary key as well as foreign keys corresponding to the Employee who received the certification and which Certification was received. The Certified entity has a one to one relationship with the Certification entity. The Relationship entity has an integer ID as its primary key. The Employee has a one to many relationship with the Relationship entity. The User entity has an integer as its primary key. There is a foreign key to the ID in the corresponding Employee record. The Employee entity has a one to one relationship with the User entity. The Message entity has a primary key which is an integer called ID. The Employee entity has a one to many relationship with the Message entity. The Group entity has an integer ID for its primary key. It also has a foreign key to the GroupMembers entity. The Group entity has a one to many relationship with the Message entity. The GroupMembers entity has a primary key which is an integer ID. The GroupMembers entity has a many to one relationship with the Group entity. The Email History has a unique integer ID as its primary key. The Employee entity has a one to many relationship with the Email History entity.

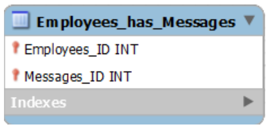
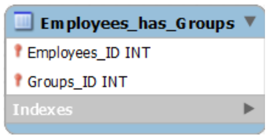
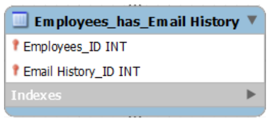
## Database Development

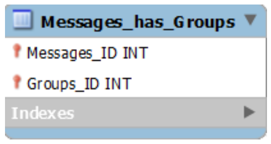
| Table Name | Code | Table in EER | Description |
|---|---|---|---|
| Employee | -- Creates Employees Table<br>CREATE TABLE Employees (<br>  ID INT NOT NULL PRIMARY KEY,<br>  Fname VARCHAR(45),<br>  Lname VARCHAR(45),<br>  email VARCHAR(45),<br>  phoneNum VARCHAR(12),<br>  WorkNum VARCHAR(12),<br>  gender CHAR(1),<br>  age INT,<br>  Department_ID INT NOT NULL,<br>  Supervisor_ID INT NOT NULL,<br>  FOREIGN KEY (Department_ID) REFERENCES Department (ID),<br>  FOREIGN KEY (Supervisor_ID) REFERENCES Supervisor (ID)<br> ); |  Employees<br>ID INT<br>Fname VARCHAR(45)<br>Lname VARCHAR(45)<br>email VARCHAR(45)<br>phonenum VARCHAR(12)<br>worknum VARCHAR(12)<br>gender CHAR(1)<br>age INT<br>Department_ID INT<br>Supervisor_ID INT<br>Indexes | The Employee table has a primary key ID. It has foreign keys Department_ID and Supervisor_ID. The Department_ID shows the "belongs to" relationship between Employee and Department. The Supervisor_ID represents the "supervises" relationship between Employee and Supervisor. We chose the VARCHAR type to represent Fname, Name, email, phoneNum, and WorkNum. The gender attribute is represented by a single CHAR. Age is represented by an INT. |
| Users | -- Creates User Table<br>CREATE TABLE users (<br>  userlogin INT NOT NULL PRIMARY KEY,<br>  userpassword VARCHAR(45)<br>,<br>  useremail VARCHAR(45) ,<br>  usertype VARCHAR(45) ,<br>  loginkey VARCHAR(45) ,<br>  Employees_ID INT NOT NULL,<br>    FOREIGN KEY (Employees_ID) REFERENCES Employees (ID)<br>    ); |  users<br>userlogin INT<br>userpassword VARCHAR(45)<br>useremail VARCHAR(45)<br>usertype VARCHAR(45)<br>loginkey VARCHAR(45)<br>Employees_ID INT<br>Indexes | The User table has userlogin, an INT, as its primary key. It has one foreign key, Employees_ID, which represents the "login" relationship between Employee and User. The VARCHAR type is used to represent the userpassword, useremail, usertype and loginkey. |

| | | | |
|---|---|---|---|
| Relationship | -- Creates Relationship Table<br>CREATE TABLE  Relationship (<br>  ID INT NOT NULL PRIMARY KEY,<br>  user1ID INT ,<br>  user2ID INT ,<br>  Rstatus VARCHAR(45) ,<br>  Rdate DATE<br>  ); |  Relationship<br>ID INT<br>user1ID INT<br>user2ID INT<br>Rstatus VARCHAR(45)<br>Rdate DATE<br>Indexes | The Relationship table has a primary key called ID which is an INT. It has two more INT type attributes: user1ID and user2ID. The status of the relationship, Rstatus, is a VARCHAR. The date, Rdate, is a DATE type. |
| Group Members | -- Creates GroupMembers Table<br>  CREATE TABLE GroupMembers (<br>  ID INT NOT NULL PRIMARY KEY,<br>  GroupID INT ,<br>  UserID INT ,<br>  JoinDate DATE<br>  ); |  GroupMembers<br>ID INT<br>GroupID INT<br>UserID INT<br>JoinDate DATE<br>Indexes | The GroupMembers table has a primary key called ID which is an INT. It has two more INT type attributes: GroupID and UserID. It also has a JoinDate attribute of type DATE. |
| Group Details | -- Creates GroupDetails Table<br>  CREATE TABLE GroupDetails (<br>  ID INT NOT NULL PRIMARY KEY ,<br>  LeaderID INT ,<br>  GroupName VARCHAR(45) ,<br>  CreatedDate DATE ,<br>  GroupMembers_ID INT NOT NULL,<br>   FOREIGN KEY (GroupMembers_ID) REFERENCES GroupMembers (ID)<br>   ); |  GroupsDetails<br>ID INT<br>LeaderID INT<br>GroupName VARCHAR(45)<br>CreatedDate DATE<br>GroupMembers_ID INT<br>Indexes | The GroupsDetails table has a primary key called ID which is an INT. There is another INT type attribute called LeaderID. The GroupDetails has a name which is represented by a VARCHAR. The creation date is DATE type. There is a foreign key called GroupMembers_ID which represents the "contains" relationship between GroupDetails and GroupMembers. |

| | | | |
|---|---|---|---|
| Messages | -- Creates Messages Table<br>CREATE TABLE Messages (<br>  ID INT NOT NULL PRIMARY KEY,<br>  senderID INT ,<br>  userID INT ,<br>  groupID INT ,<br>  Messagedate DATE<br>  ); |  Messages<br>ID INT<br>senderID INT<br>userID INT<br>groupID INT<br>date DATE<br>Indexes | The Messages table has a primary key called ID of type INT. It also has senderID, userID, and groupID which are all INT types. It also has a date attribute of type DATE. |
| Email History | -- Creates EmailHistory Table<br> CREATE TABLE EmailHistory<br>(<br>  ID INT NOT NULL PRIMARY KEY,<br>  SenderID INT ,<br>  Message LONGTEXT ,<br>  ReceiverID INT ,<br>  EmailDate DATE<br>  ); |  Email History<br>ID INT<br>SenderID INT<br>Message LONGTEXT<br>ReceiverID INT<br>EmailDate DATE<br>Indexes | The EmailHistory table has a primary key called ID of type INT. It has a SenderID and ReceiverID which are also INT types. The message is LONG TEXT type so that the size limit is very large. The EmailDate is a DATE type. |
| Certified | -- Creates Certified Table<br>CREATE TABLE Certified (<br>  ID INT NOT NULL PRIMARY KEY,<br>  UserID INT ,<br>  CertificationID INT ,<br>  Date DATE ,<br>  Employees_ID INT NOT NULL,<br> FOREIGN KEY (Employees_ID)<br>REFERENCES Employees (ID)<br>    ); |  Certified<br>ID INT<br>UserID INT<br>CertificationID INT<br>Date DATE<br>Employees_ID INT<br>Indexes | The Certified table has a primary key called ID of type INT. It has a UserID and CertificationID which are INT types. It also has a Date which is a DATE type. There is a foreign key called Employees_ID which represents the “are” relationship between Certified and Employee. |
| Certification | -- Creates Certification Table<br>  CREATE TABLE Certification (<br>  ID INT NOT NULL PRIMARY KEY,<br>  Name VARCHAR(45) ,<br>  Type VARCHAR(45) ,<br>  Certified_ID INT NOT NULL,<br>  FOREIGN KEY (Certified_ID)<br>REFERENCES Certified(ID)<br>   ); |  Certification<br>ID INT<br>Name VARCHAR(45)<br>Type VARCHAR(45)<br>Certified_ID INT<br>Indexes | The Certification table has a primary key called ID of type INT. The VARCHAR type is used to represent the Name and Type attributes. There is a foreign key called Certified_ID which represents the “are” relationship between Certified and Certification. |

| | | | |
|---|---|---|---|
| Supervisor | -- Creates Supervisor Table<br>CREATE TABLE Supervisor (<br>ID INT NOT NULL PRIMARY KEY,<br>UserID INT,<br>DepartmentID INT<br>); |  Supervisor<br>ID INT<br>UserID INT<br>DepartmentID INT<br>Indexes | The Supervisor table has a primary key called ID of type INT. It has a UserID and DepartmentID which are both INTs. |
| Department | -- Creates Department Table<br>CREATE TABLE Department (<br>ID INT NOT NULL PRIMARY KEY,<br>DName VARCHAR(45),<br>Supervisor_ID INT NOT NULL,<br>FOREIGN KEY (Supervisor_ID) REFERENCES Supervisor (ID)<br>); |  Department<br>ID INT<br>DName VARCHAR(45)<br>Supervisor_ID INT<br>Indexes | The Department table has a primary key called ID which is an INT. It has a name which is represented by a VARCHAR. It has a foreign key called Supervisor_ID. |
| Employees_has_Relationship | -- Creates Employees_has_Relationship Table<br>CREATE TABLE Employees_has_Relationship (<br>Employees_ID INT NOT NULL ,<br>Relationship_ID INT NOT NULL ,<br>PRIMARY KEY (Employees_ID, Relationship_ID),<br>FOREIGN KEY (Employees_ID) REFERENCES Employees (ID),<br>FOREIGN KEY (Relationship_ID) REFERENCES Relationship(ID)<br>); |  Employees_has_Relationship<br>Employees_ID INT<br>Relationship_ID INT<br>Indexes | The Employees_has_Relationship table is used to represent the many to many relationship between the Employee and Relationship entities. Therefore it has two foreign keys: Employees_ID and Relationship_ID. These foreign keys together make up the primary key for the table. |

| | | | |
|---|---|---|---|
| Employees_has_Messages | -- Creates Employees_has_Messages Table<br>CREATE TABLE Employees_has_Messages (<br>Employees_ID INT NOT NULL,<br>Messages_ID INT NOT NULL,<br>PRIMARY KEY (Employees_ID, Messages_ID),<br>FOREIGN KEY (Employees_ID) REFERENCES Employees (ID),<br>FOREIGN KEY (Messages_ID) REFERENCES Messages (ID)<br>); |  Employees_has_Messages<br>Employees_ID INT<br>Messages_ID INT<br>Indexes | The Employees_has_Messages table is used to represent the many to many relationship between the Employee and Message entities. Therefore it has two foreign keys: Employees_ID and Messages_ID. These foreign keys together make up the primary key for the table. |
| Employees_has_Groups | -- Creates Employees_has_Groups Table<br>CREATE TABLE Employees_has_Groups (<br>Employees_ID INT NOT NULL,<br>Groups_ID INT NOT NULL,<br>PRIMARY KEY (Employees_ID, Groups_ID),<br>FOREIGN KEY (Employees_ID) REFERENCES Employees(ID),<br>FOREIGN KEY (Groups_ID) REFERENCES GroupDetails(ID)<br>); |  Employees_has_Groups<br>Employees_ID INT<br>Groups_ID INT<br>Indexes | The Employees_has_Groups table is used to represent the many to many relationship between the Employee and GroupDetails entities. Therefore it has two foreign keys: Employees_ID and Groups_ID. These foreign keys together make up the primary key for the table. |
| Employees_has_Email_History | -- Creates Employees_has_Email_History Table<br>CREATE TABLE Employees_has_Email_History (<br>Employees_ID INT NOT NULL,<br>Email_History_ID INT NOT NULL,<br>PRIMARY KEY (Employees_ID, Email_History_ID),<br>FOREIGN KEY (Employees_ID) |  Employees_has_Email History<br>Employees_ID INT<br>Email History_ID INT<br>Indexes | The Employees_has_EmailHistory table is used to represent the many to many relationship between the Employee and EmailHistory entities. Therefore it has two foreign keys: Employees_ID and Email_History_ID. These foreign keys together |

| | | | |
|---|---|---|---|
| | REFERENCES Employees(ID),<br>FOREIGN KEY (Email_History_ID) REFERENCES EmailHistory (ID)<br>); | | make up the primary key for the table. |
| Messages_has_Groups | -- Creates Messages_has_Groups Table<br>CREATE TABLE Messages_has_Groups (<br>Messages_ID INT NOT NULL,<br>Groups_ID INT NOT NULL,<br>PRIMARY KEY (Messages_ID, Groups_ID),<br>FOREIGN KEY (Messages_ID) REFERENCES Messages (ID),<br>FOREIGN KEY (Groups_ID) REFERENCES GroupDetails (ID)<br>); |  Messages_has_Groups<br>Messages_ID INT<br>Groups_ID INT<br>Indexes | The Messages_has_Groups table is used to represent the many to many relationship between the Message and GroupDetails entities. Therefore it has two foreign keys: Messages_ID and Groups_ID. These foreign keys together make up the primary key for the table. |

## References

[1] Fuchs, Jay. “The 14 Best Contact Management Software Tools in 2022.” HubSpot Blog, HubSpot, 2 Sept. 2022, https://blog.hubspot.com/sales/contact-management-software.

[2] Keap, 2022, https://keap.com/features/client-management.

[3] Scheiner, Michael. “Keap CRM Review: Keap Software Prices, Features, Pros & Cons.” CRM.org, 23 May 2022, https://crm.org/news/keap-crm-review.

[4] Scheiner, Michael. “Streak CRM Review: Pricing, Pros & Cons of Streak for Gmail Chrome Extension.” CRM.org, 8 Feb. 2022, https://crm.org/news/streak-crm-review.

[5] Singleton, Chris. “Nimble CRM Review (2022) - Full Pros and Cons.” Style Factory, 25 Jan. 2022, https://www.stylefactoryproductions.com/blog/nimble-crm-review.

[6] “DBMS Keys: Primary, Foreign, Candidate and Super Key - Javatpoint.” www.javatpoint.com, www.javatpoint.com/dbms-keys. Accessed 3 Oct. 2022.

[7] “Primary and Foreign Key Constraints - SQL Server.” Microsoft Learn, 17 Aug. 2022, learn.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-ver16.

[8] Techopedia. “Relationship.” Techopedia.com, 23 Sept. 2014, www.techopedia.com/definition/24438/relationship-databases#:%7E:text=A%20relationship%2C%20in%20the%20context,while%20linking%20disparate%20data%20items.