Contact Management System

CMPT 308N-113

11161

The Data Bros

Marist College School of Computer Science and Mathematics

Shane Seeley (Team Leader)

Email: shane.seeley1@marist.edu

Jozef Maselek

Email: jozef.maselek1@marist.edu

GitHub: https://github.com/ShaneSeeley28/CMPT308N-113_Contact_Management_System_DataBros

Date: 9/26/2023

Table of Contents

Section 1

Why We Make a Great Team

Jozef's Paragraph:

I want to work with Shane because I know he is a student who makes good decisions on essential requirements. I know that he can lead our team to meet project conditions and submit work on time. Shane is a good leader due to his tenacity in doing every task assigned to him. Shane is also a hard worker and I know that he will submit our projects on time and get his work done effectively.

Shane's Paragraph:

I am excited to work with Jozef because I know we can get our work done on time and answer questions efficiently. We both are passionate about computer science and learning about topics like databases. The reason for me being the team leader is due to my knowledge of SQL and my personality, which will help guide our team to being as effective as possible.

Different Contact Management Systems:

Cloud-Based Contact Management System "Google Contacts":

Advantages: Contacts are accessible from any device with internet access, and data is backed up regularly, which reduces the risk of data loss (1).

Disadvantages: Requires an internet connection to access data, which can create limitations (1).

CRM Systems "Salesforce":

Advantages: Automation CRM systems automate specific tasks like email campaigns and customer interactions using stored data. (2)

Disadvantages: CRM systems can be expensive and may not be needed for small businesses. (2).

Spreadsheet Software "Excel":

Advantages: allows for flexibility with data organization and sorting, data can also be integrated with other software using Excel. (3)

Disadvantages: Version control issues collaboration in Excel can be challenging for data entry. (3)

Why our Software is the right choice and Our Goals.

We believe that our system will be cost-effective and perfect for small businesses looking for a Contact Management System. We will create a secure SQL database to store customers' information that users can easily access with a state-of-the-art user interface. Our system will have up-to-date security making it perfect for storing user data. With our Contact Management System users can get the best possible CMS for a price that's affordable.

Section 2

Describe how you created this mini world and how you selected the entities, attributes, relationships, participations, and cardinality.

To create this mini world, we thought about what would be important for a contact management system to function and what kind of amenities users might want. Because it's acontact-focusedd database the entities are centered around the contacts entity. Attributes belonging to each entity are self-explanatory and are indicative of the table they are assigned to. The entities not directly related to the contacts entity are meant to further elaborate and give insight on information like departments or notes on interactions. Participations are tied together by necessity for the most part.

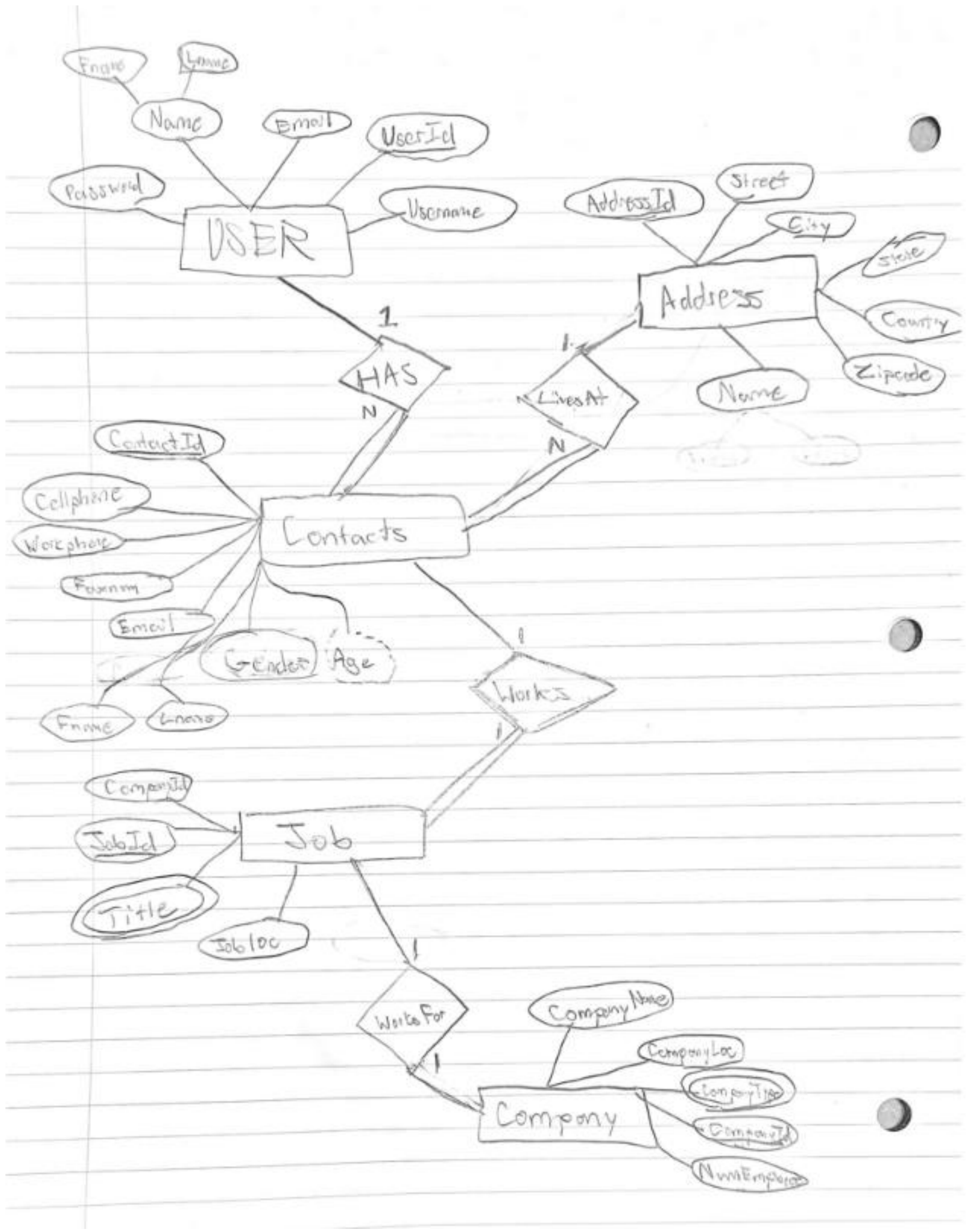


Figure 1 - User ER model

User ER Model - Provide a short description of each entity, attribute, relationship, participation, and cardinality.

Address is an entity that stores an AddressID, the street address, city, state, country, zip code, and the name of the person associated. It is related to the contacts entity and stores the address information of specific contacts. Many contacts can store many addresses.

Contacts is an entity with the attributes Fname, Lname, Cellphone, Workphone, FaxNum, Email, Gender, and Birthday. It stores the contact info for a person. It's related to most of the other entities in many different ways.

User is an entity with the attributes Username, Password, UserID, IsAdmin, Email, Fname, and Lname. It is an entity to store the user’s information and has an attribute to discern whether it's an admin. Its connected to contacts in a 1:N configuration.

Job, Company, and Department are related entities, describing the jobs of individual contacts, the company they work for, and the department they work in that company. Job has the attributes JobID, Title, JobLocation, CompanyID, and ContactID. It is connected to contacts and company in a 1:1 relationship. Company has CompanyID, CompanyName, CompanyLocation, CompanyType, and Employees and is connected to Job and Department in a 1:1 relationship. Department is connect to Company and Contacts in a 1:1 relationship and has the attributes DepartmentID, Dname, Dlocation, DcompanyID, Demployees, and Dmanagername.

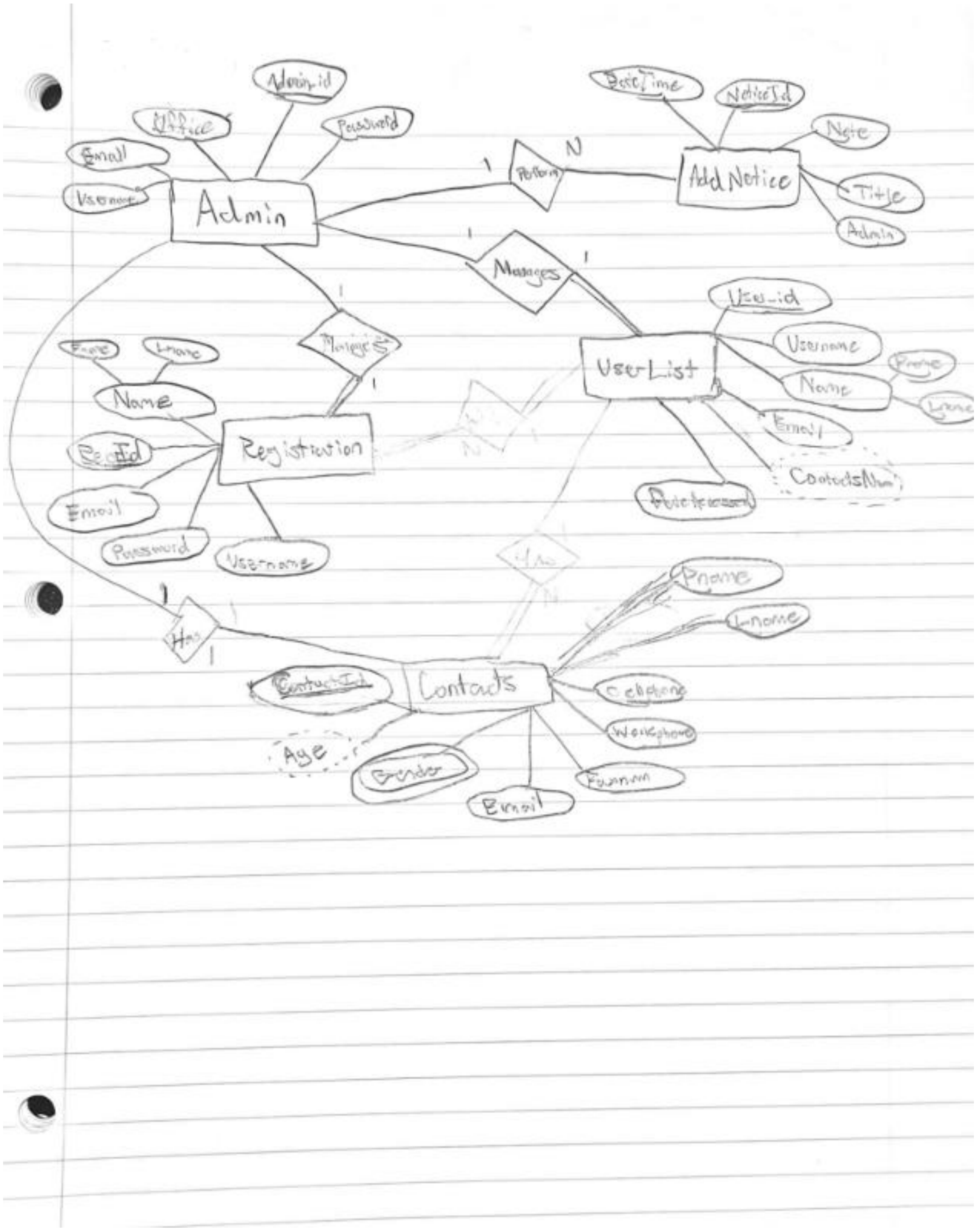


Figure 2 - Admin ER model

Admin ER Model - Provide a short description of each entity, attribute, relationship, participation, and cardinality.

Contacts is an entity with the attributes Fname, Lname, Cellphone, Workphone, FaxNum, Email, Gender, and Birthday. It stores the contact info for a person. Its function in this ER model is in the case an admin wants to access a list of Admin-related contacts.

Admin is an entity with the attributes Name, Admin_ID, Password, Email, and Username. It stores the info of Admins for the database system. It connects to all of the other entities other than contacts.

UserList is an entity with the attributes User_ID, username, name, email, contactsNum, and Date accessed. It stores the list of Users in the database and allows admins to change or view that information. It has a 1:1 relationship with Registration.

Registration is an entity with the attributes Name, RegID, Email, Password, and Username. This entity stores the list of users that can be added but aren't yet. The admin would have to go through and approve of the users in this list to be added to UserList. It has a 1:1 relationship with Admin.

AddNotice is an entity with the attribute DateTime, NoticeID, Note, Title, and Admin. This is an entity that stores the information for notices about the Contact management system. It has a 1:N relationship with Admin as an admin could have many notices.

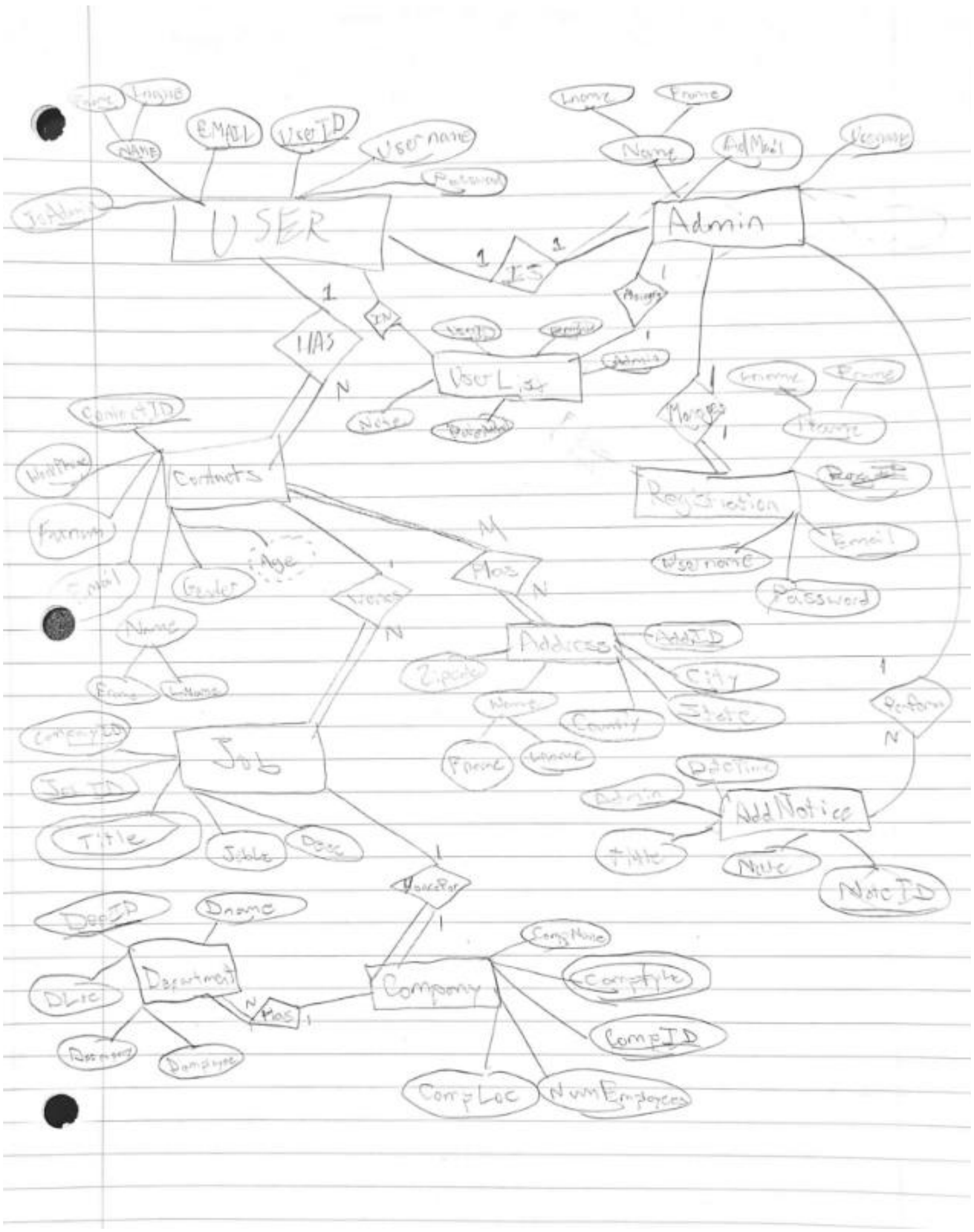


Figure 3 - Conceptual ER Model

Description of Conceptual Model

This conceptual model provides a high-level view of the key elements and relationships within the Contact Management System, serving as a foundation for developing the system's detailed design and implementation. It helps people understand the system's structure and functionality without delving into technical specifics. The model provides a basic view of what the contact management system will end up looking like.

Interaction and the Notes entities are the big new entities. They are meant to keep tabs on interactions made between contacts, such as calls, and to keep notes on the interaction. Interaction has the attributes InteractionID, Date, Itype, AssociatedContact, and Company. It is connected to contacts as many to one contact. Notes has noteID, Ntype, NOTE, Date, and InteractionID. It is connected to Interactions in 1:1 and is meant to keep track of the interactions individually.

Relationships is an entity that is connected 1:1 to Contacts. It has the attributes RelationshipID, Fname, Lname, Rtype, Fname2, Lname2. It is an entity meant to keep track of personal relationships between people. Groups is a similar entity that keeps track of group relationships rather than personal relationships. Groups has the attributes GroupID, GroupName, Gmembers, Gcompany, and Gtype. It is connected to contacts in M:N.

The rest of the model just combines the two pre-existing ER models into one conceptual model. Therefore, it is the same as the other two.

Provide a short description of keys and relationships.

Every entity has a key with most having it connected to another entity through a relationship. All of them are along the lines of "EnitityID," such as AdminID, UserID, ContactID, or JobID. Entities are typically related in the way of being able to modify another. Users can add contact info to contacts such as addresses, jobs, companies, and so on. Admins can add registration info to users and can modify their contacts and so on.

Describe how you implemented these features on your EER model.

I implemented these features accordingly into my EER model. The way I designed the contact management system meant the Conceptual and EER models are fairly similar. So for users and for the admin a lot of their capabilities line up similarly for them to come together in the EER model. One big change is that I moved the Admin functionalities to be an attribute of the user entity just to make the model a little tidier.

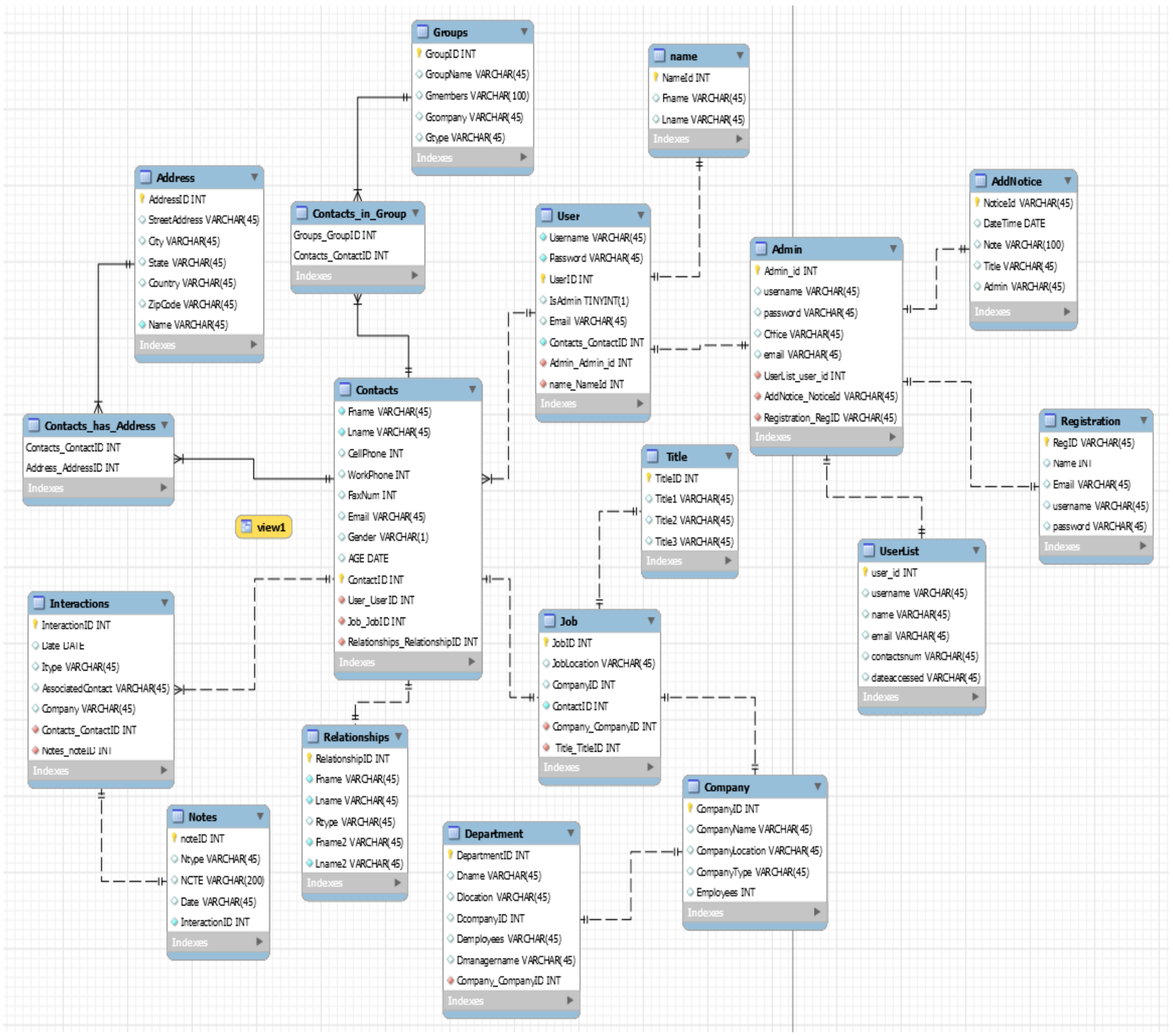


Figure 4 - EER Diagram

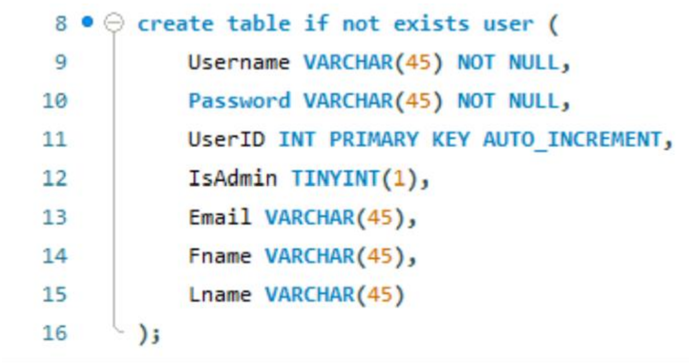
Description of EER Diagram

Our EER diagram is a realization of our ER models and our conceptual model. It extends the capabilities of the ER model to a more usable form. It better represents the complex structure, inheritances, and specializations of our contact management system. It allows us to consider more intricate and precise relationships and hierarchies for our system. The EER diagram provides a structured representation of how contact data moves around our database. It ultimately provides a blueprint for our group to create and support a Contact Management database. It closely follows the conceptual diagram for the most part, aside from the absence of the Admin table. It has all of the same entities and provides the same relationships for the most part.

Section 3

Implementation of Our Database

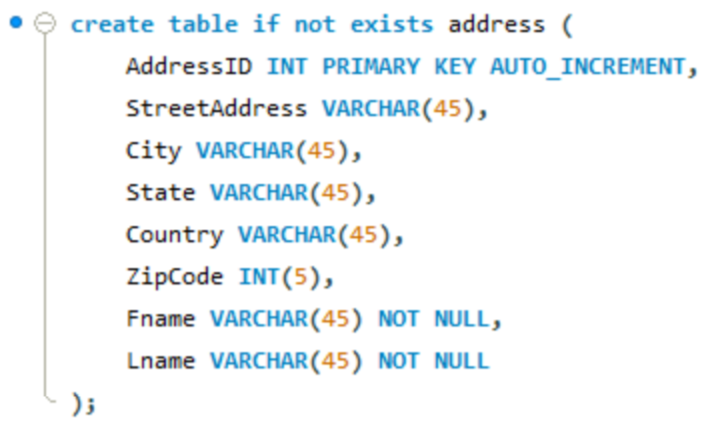
The user table has 7 attributes. Username and password are both not null varchars, UserID is the primary key and auto increments, IsAdmin is a tinyINT of 1 essentially working like an off/on switch, and then email, Fname, and Lname are all varchars. This user closely aligns with our ER Model's version of user and combines with the admin model by having the IsAdmin attribute. Every varchar datatype in this database is capped at (45) so as to save space without compromising unless otherwise specified.



```sql
create table if not exists user (
    Username VARCHAR(45) NOT NULL,
    Password VARCHAR(45) NOT NULL,
    UserID INT PRIMARY KEY AUTO_INCREMENT,
    IsAdmin TINYINT(1),
    Email VARCHAR(45),
    Fname VARCHAR(45),
    Lname VARCHAR(45)
);
```

Figure 5 - table for user

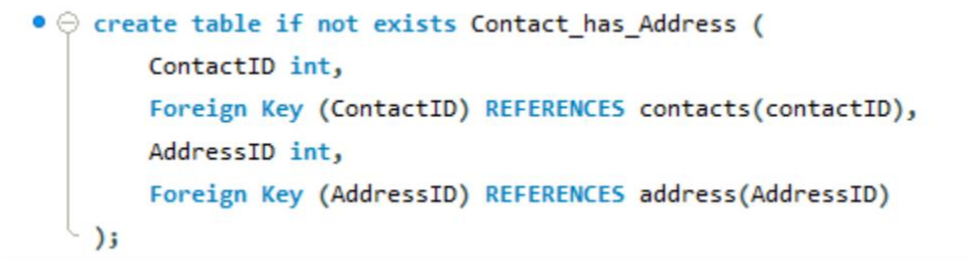
The address table has 8 attributes. AddressID is an INT primary key and auto increments, street address, city, state, and country are all varchars, zipcode is an int capped at 5 digits like real zip-codes, and then Fname and Lname are both not null varchars. Address almost exactly resembles its form in the ER model. It also has a many-to-many relationship with contacts, as many contacts can have many addresses, and is connected via the table contact_has_address.



```
create table if not exists address (
    AddressID INT PRIMARY KEY AUTO_INCREMENT,
    StreetAddress VARCHAR(45),
    City VARCHAR(45),
    State VARCHAR(45),
    Country VARCHAR(45),
    ZipCode INT(5),
    Fname VARCHAR(45) NOT NULL,
    Lname VARCHAR(45) NOT NULL
);
```

Figure 6 - table for address

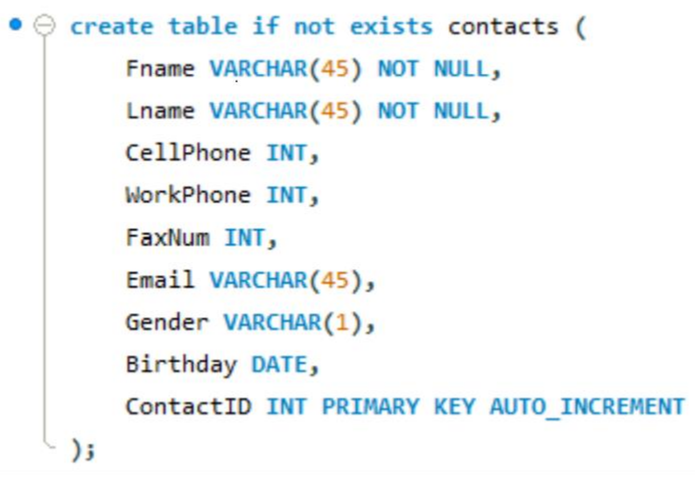
The contact_has_address table is a pass-through table for contacts and address to have a relationship. It has 2 attributes, both foreign keys from contacts and address, ContactID and AddressID. This is a new table that has to be introduced for the database to work but takes the place of one of the relationships in the ER model.



```
create table if not exists Contact_has_Address (
    ContactID int,
    Foreign Key (ContactID) REFERENCES contacts(contactID),
    AddressID int,
    Foreign Key (AddressID) REFERENCES address(AddressID)
);
```

Figure 7 - table for contact_has_address

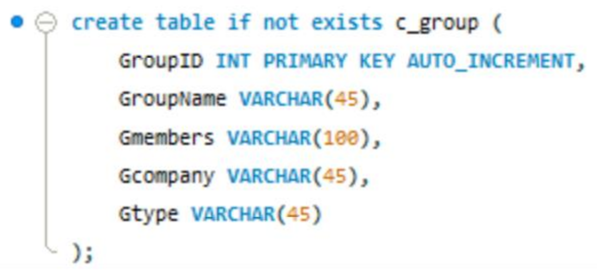
The contacts table has 9 attributes. Fname and Lname are both not null varchars, CellPhone, WorkPhone, and FaxNum are all ints, Email and Gender are both varchars but gender is a 1 char varchar, and ContactID is an int primary key that auto_increments. This table is very similar to the one that exists on the ER model and EER diagram.



```sql
create table if not exists contacts (
    Fname VARCHAR(45) NOT NULL,
    Lname VARCHAR(45) NOT NULL,
    CellPhone INT,
    WorkPhone INT,
    FaxNum INT,
    Email VARCHAR(45),
    Gender VARCHAR(1),
    Birthday DATE,
    ContactID INT PRIMARY KEY AUTO_INCREMENT
);
```

Figure 8 - table for contacts

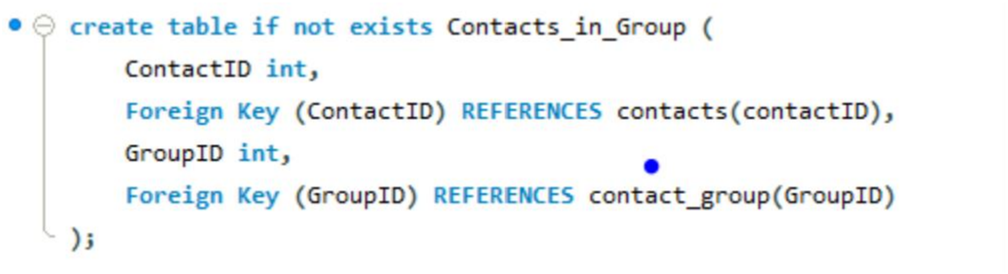
The C_group table has 5 attributes. GroupID is an int primary key that auto increments, GroupName, Gmembers, Gcompany, and Gtype are all varchars and Gmembers has 100 chars to accommodate a lot of text. It was originally called just group but mySQL does not like that word being used for a table name. It has a relationship with contacts through the contacts_in_group table. It is similar to the group entity in our ER model.



```sql
create table if not exists c_group (
    GroupID INT PRIMARY KEY AUTO_INCREMENT,
    GroupName VARCHAR(45),
    Gmembers VARCHAR(100),
    Gcompany VARCHAR(45),
    Gtype VARCHAR(45)
);
```

Figure 9 - table for C_group

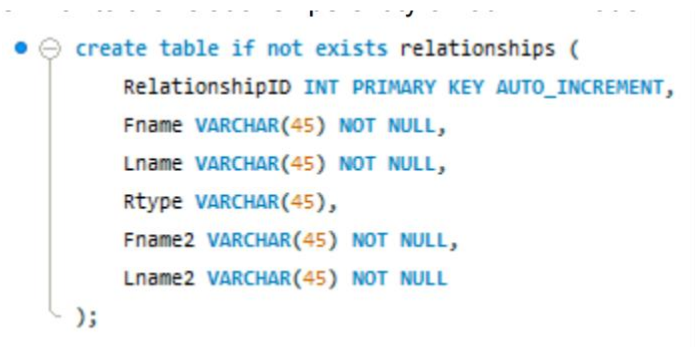
Contacts_in_group is a pass-through table to allow Contacts and group to have a many-to-many relationship. It has 2 foreign key attributes, ContactID and GroupID, which it gets from Contacts and C_group. It is in place for the relationship between the two tables in the ER model.



```
create table if not exists Contacts_in_Group (
    ContactID int,
    Foreign Key (ContactID) REFERENCES contacts(contactID),
    GroupID int,
    Foreign Key (GroupID) REFERENCES contact_group(GroupID)
);
```

Figure 10 - table for contacts_in_group

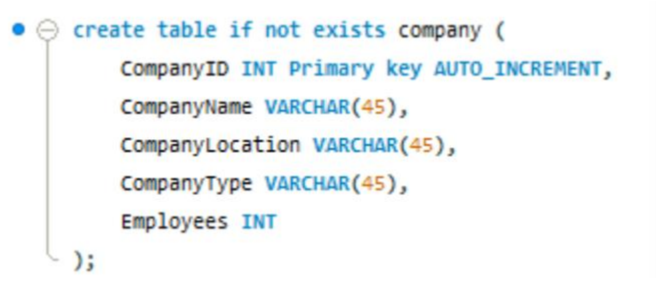
The relationships table has 6 attributes. RelationshipID is an int primary key that auto increments, and Fname, Lname, Rtype, Fname2, and Lname2 are all varchars. This table is similar to the entity of the relationship on our ER model.



```
create table if not exists relationships (
    RelationshipID INT PRIMARY KEY AUTO_INCREMENT,
    Fname VARCHAR(45) NOT NULL,
    Lname VARCHAR(45) NOT NULL,
    Rtype VARCHAR(45),
    Fname2 VARCHAR(45) NOT NULL,
    Lname2 VARCHAR(45) NOT NULL
);
```

Figure 11 - table for relationships

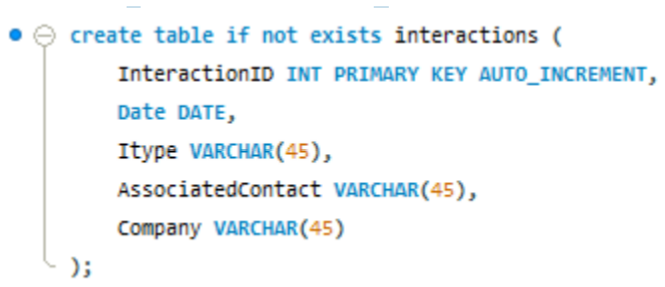
The company table has 5 attributes. CompanyID is an int primary key that auto increments, CompanyName, CompanyLocation, and CompanyType are all varchars, and Employees is an int. This table is very similar to the one on our ER model but made more practical for database usage.



```
create table if not exists company (
    CompanyID INT Primary key AUTO_INCREMENT,
    CompanyName VARCHAR(45),
    CompanyLocation VARCHAR(45),
    CompanyType VARCHAR(45),
    Employees INT
);
```

Figure 12 - table for company

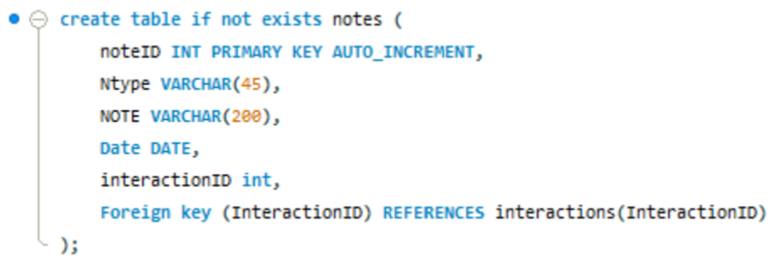
The interactions table has 5 attributes. InteractionID is an int primary key that auto increments, Date is a date, and Itype, AssociatedContact, and company are all varchars. This table differs a little from our ER model as it should be connected to company and contacts but is not to keep the database simpler.



```
create table if not exists interactions (
    InteractionID INT PRIMARY KEY AUTO_INCREMENT,
    Date DATE,
    Itype VARCHAR(45),
    AssociatedContact VARCHAR(45),
    Company VARCHAR(45)
);
```

Figure 13 - table for interactions

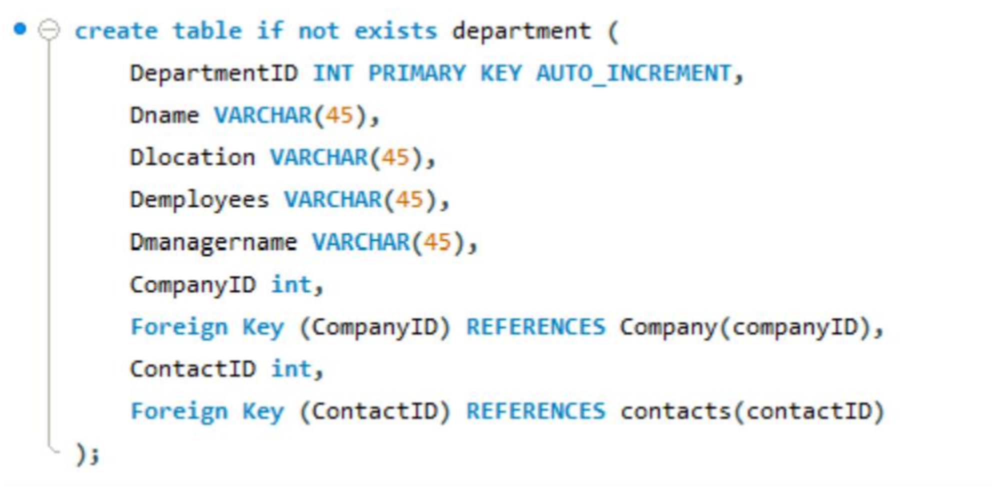
The notes table has 5 attributes. NoteID is an int primary key that auto increments, Ntype and NOTE are both Varchars and NOTE can accept up to 200 characters, date is a date, and interactionID is an int that is a foreign key from the interactions table. This way notes are a derivative of interactions and are connected to an interaction.



```
create table if not exists notes (
    noteID INT PRIMARY KEY AUTO_INCREMENT,
    Ntype VARCHAR(45),
    NOTE VARCHAR(200),
    Date DATE,
    interactionID int,
    Foreign key (InteractionID) REFERENCES interactions(InteractionID)
);
```

Figure 14 - table for notes

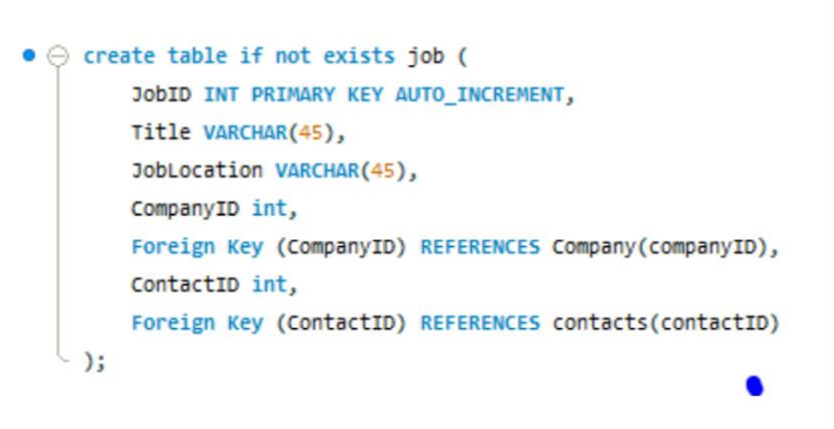
The department table has 7 attributes. DepartmentID is an int primary key that auto increments, Dname, Dlocation, Demployees, and Dmanagername are all varchars, and CompanyID and ContactID are both foreign keys from the Company and Contacts tables. They allow contacts and company to have a 1:1 relationship each with department. The table is very similar to the entity found in the ER models.



```
create table if not exists department (
    DepartmentID INT PRIMARY KEY AUTO_INCREMENT,
    Dname VARCHAR(45),
    Dlocation VARCHAR(45),
    Demployees VARCHAR(45),
    Dmanagername VARCHAR(45),
    CompanyID int,
    Foreign Key (CompanyID) REFERENCES Company(companyID),
    ContactID int,
    Foreign Key (ContactID) REFERENCES contacts(contactID)
);
```

Figure 15 - table for departments

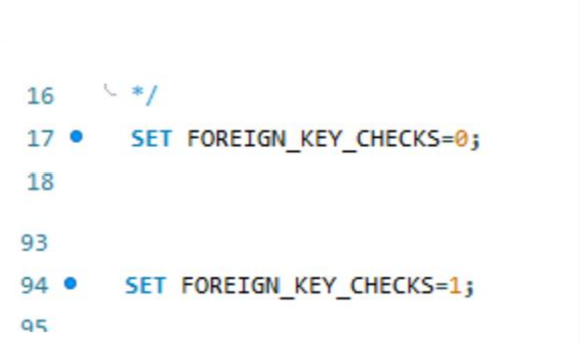
The job table has 5 attributes. JobID is an int primary key that auto increments, title and JobLocation which are both Varchars, and CompanyID and ContactID which are both int foreign keys from the company and contact's table. This way the Job table can have a 1:1 relationship with contacts and a 1:M relationship with company.



```
create table if not exists job (
    JobID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(45),
    JobLocation VARCHAR(45),
    CompanyID int,
    Foreign Key (CompanyID) REFERENCES Company(companyID),
    ContactID int,
    Foreign Key (ContactID) REFERENCES contacts(contactID)
);
```

Figure 16 - table for jobs

Section 4:

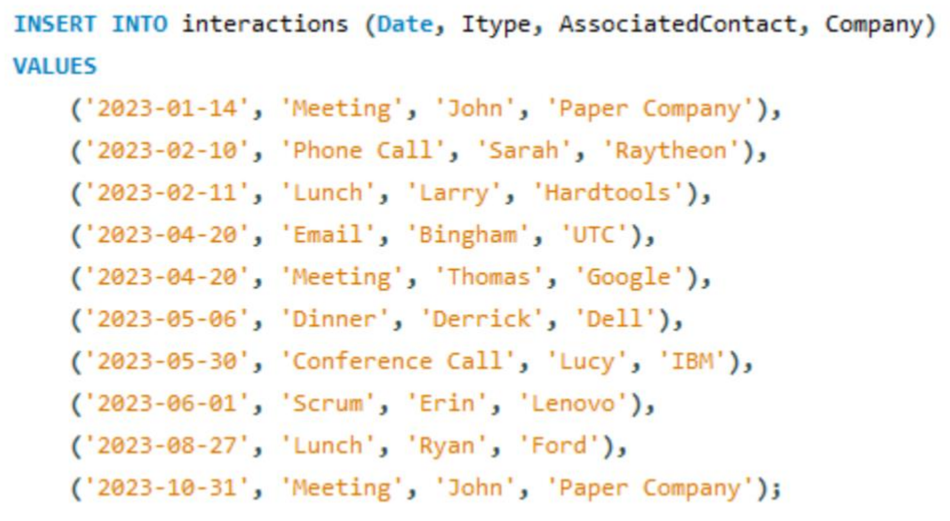


```
*/
SET FOREIGN_KEY_CHECKS=0;


SET FOREIGN_KEY_CHECKS=1;
```

Figure 17 - set foreign key statements

To handle foreign constraints we simply set FOREIGN_KEY_CHECKS to be equal to 0. This turns off MySQL's capability to check for foreign keys and allows us to add data to tables with disregard to them. This is a very simple way of adding data while circumventing the errors that may happen due to foreign keys. It is also much faster than removing the foreign keys attribute-by-attribute in each of the tables, then adding data, and then re-adding the foreign keys. This is a simple and effective way of side stepping the issues of foreign keys.



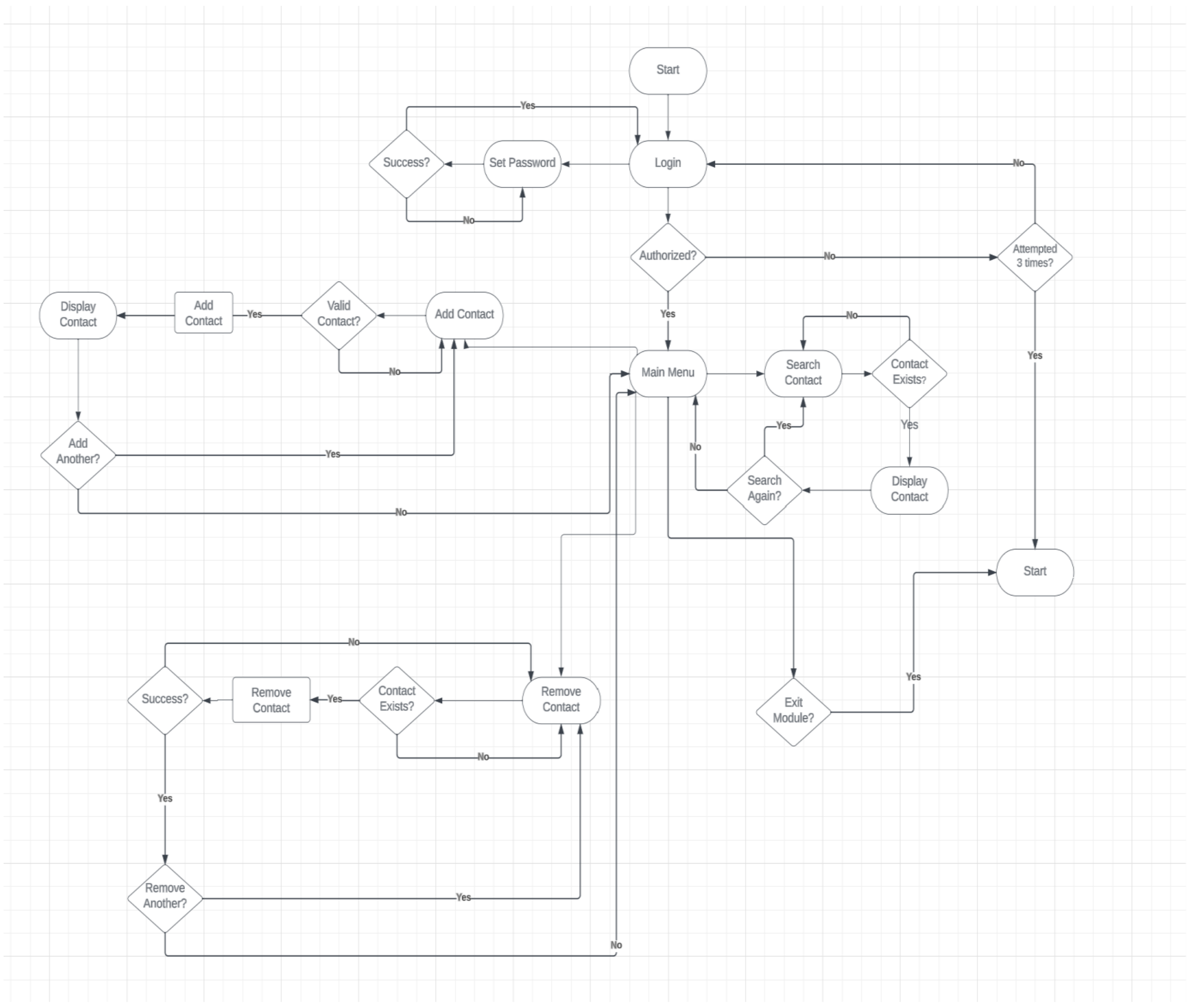
```
INSERT INTO interactions (Date, Itype, AssociatedContact, Company)
VALUES
    ('2023-01-14', 'Meeting', 'John', 'Paper Company'),
    ('2023-02-10', 'Phone Call', 'Sarah', 'Raytheon'),
    ('2023-02-11', 'Lunch', 'Larry', 'Hardtools'),
    ('2023-04-20', 'Email', 'Bingham', 'UTC'),
    ('2023-04-20', 'Meeting', 'Thomas', 'Google'),
    ('2023-05-06', 'Dinner', 'Derrick', 'Dell'),
    ('2023-05-30', 'Conference Call', 'Lucy', 'IBM'),
    ('2023-06-01', 'Scrum', 'Erin', 'Lenovo'),
    ('2023-08-27', 'Lunch', 'Ryan', 'Ford'),
    ('2023-10-31', 'Meeting', 'John', 'Paper Company');
```

Figure 18 - a multi-valued insert into statement

The best optimization practice is to use the LOAD DATA INFILE query. However, after some testing, that query doesn’t ever work correctly, instead giving various errors. So, in its stead, we decided to instead go with the next best option available to us, compound insert into statements. This is considerably faster (many times faster in some cases) than using separate single-row INSERT statements. However, LOAD DATA is still much faster than using these insert into statements. LOAD DATA is usually 20 times faster than using INSERT statements. That's 2000% faster. However, due to errors in our work, this method does not work properly.

Inserting the data in one query through a multi-value insert statement, like the one shown in Figure 18, only took 0.016 sec. This applies all the data at once. Alternatively, doing a single insert statement for each piece of data takes 0.016 seconds to accomplish. To insert ten different statements would therefore take 0.16 seconds. This means that the multi-valued insert statement is 10x faster than the ten single insert statements.

Section 5:



Login Page: Gives access to Set Password for account creation and setting a new password for your username.

Input: Username and password

Output: Valid username and password enable the end user access to the

main menu. Invalid username or password cause showing a warning message of invalid username and password. After 3 attempts the user will be sent back to the start and forced to create new password

Set Password Page:

Input: Password entered twice for confirmation

Output: Success updates the end user's password for login and displays that the password has been updated for the end user, failure redirects the end user back to the set password page.

Main Menu Page: Grants the end user access to Search Contact Page, Add Contact Page, Remove Contact Page, and access to exit the database application.

Input: Search Contact, Add Contact, Remove Contact, Exit.

Output: Search Contact -> redirects to Search Contact Page, Add Contact -> redirects to Add Contact Page, Remove Contact -> redirects to Remove Contact Page, Exit -> exits application redirects to login page.

Add Contact Page:

Input: Contact information userId, isAdmin, email, Fname, Lname.

Output: if the information entered is valid the contact will be added and the end user will be redirected to the Display Contact page where all contact information is displayed userid, isAdmin, email, Fname, Lname. The end user will then be asked if would like to add another contact if yes, they will be redirected back to the Add Contact Page, if no they will be redirected back to the main menu.

Remove Contact Page:

Input: userId

Output: if the information entered is a valid contact the contact will be removed and the end user will be shown that the contact has been removed, the end user will then be asked if they would like to remove another contact if yes they will be redirected back to the Remove Contact Page, if no they will be redirected back to the main menu.

Search Contact Page:

Input: userId

Output: the program will check if the user exists, the end user will be redirected to the Display Contact Page where they will see the userId, isAdmin, email, Fname, Lname of the userId that they just entered, then they will be asked if they would like to search another user if yes they will be redirected back to the search contact page, if no then they will be redirected to the main menu.

References:

(1) "Google Contacts." *Google Contacts Software Reviews, Demo & Pricing - 2023*, www.softwareadvice.com/crm/google-contacts-profile/. Accessed 25 Oct. 2023.
(2) "Customer Relationship Management Definition - Salesforce Us." *Salesforce*, www.salesforce.com/crm/what-is-crm/. Accessed 25 Oct. 2023.
(3) "WhatCMS - Microsoft Excel." *What CMS?*, whatcms.org/c/Microsoft-Excel. Accessed 25 Oct. 2023.