Edgar Carbajal
CSCI 466-2
Due: 3/1/2023

ER Diagram, to Relational DB, to SQL
Assignment #5

This is the PDF for the explanation/choices on what was included for the ER Diagram from the prompt in Assignment #5, and the conversion of this diagram to Relational DB form, where this form will be used to write SQL queries to create Relational DB in MariaDB.

The first section of this PDF, will be the choices for the ER Diagram, the second section will be the conversion to Relational DB format, and in a separate text file will be the queries/code written to create the Relational DB representation in a DBMS such as MariaDB.

ER Diagram Choices/Explanations

This is where the entities, relationships, and attributes chosen for the ER Diagram to serve as a database for the problem given in the prompt will be explained, similar to Assignment #2. All of these choices were made in order to satisfy all of the requirements given by the prompt.

Entities

- Employee
Chosen as an entity as it relates to the employees of the Spa enterprise, in which they interact with other potential entities such as Clients, and are more likely going to have different number/types of employees, and also have a multitude of information needed to store with them as attributes. The attribute list for employees is large: FName, LName, Address, Email, Phone, PayRate, JobTitle, JobSpec, and as a primary key, EmpID which is a surrogate key meant to make it easy to identify a particular employee in the database.

- Client
Chosen as an entity as it relates to the potential clients of the Spa enterprise, in which they interact with other potential entities such as Appointments, and are more likely going to have different number/types of clients with multitude of different information which they are stored in the attributes: FName, LName, Address, Email, Phone, and ClientID as a surrogate primary key for easy identification in the database.

- Appointment
Chosen as an entity as it is the intermediary interaction between the clients, and the services/employees that the enterprise provides, while also most likely having more than one possible entry per client thus, it needs to be an entity. The only attribute included for this entity in the diagram is the surrogate primary key: AppID. No other attributes were added since everything else about the appointment is either handled by other entities or intersection data. Specifically the Date & Time aspect since a Client can theoretically reserve the same appointment more than once, but at different dates of course, then having a separate Date & Time entity helps alleviate any issues when transforming our ER Diagram to Relational DB form.

Edgar Carbajal
CSCI 466-2
Due: 3/1/2023

- Service
Chosen as an entity since it will interact with the Appointment and Employee entities and because it inherently has multiple different types since there is multiple different services to choose from in the Spa given by the prompt, and can be used to expand to more services easily in the future using an entity format. The two attributes are the primary key: Type, which refers to the type of service offered (hair, makeup, manicures, massages, and waxes), and Price which refers to the price of each service.

- Date & Time
Chosen as an entity because for the relationship between Client and Appointment, there will be multiple Dates & Times that a client might reserve the same kind of appointment and can cause problems when translating the ER Diagram into a Relational DB in 3NF, so to not cause any issues, it was made into an entity, but will be later be places as an attribute according to 3NF rules. The attributes/primary key are: Datetime.

- Shift
Chosen as an entity because for the relationship between Employee and Service as an employee works a shift which is a set amount of time at a date in which they work in. This entity basically acts as a Date entity, so we might have to take the same approaches when transforming it to Relational DB. The attributes/primary key are: Date, StartTime, and EndTime.

Relationships

*
- reserves
A relationship that links Appointment, Client, and Date & Time entities toghether. Chosen as a relationship to represent when a client decides to make/reserve an appointment in the Spa. It has intersection data named: PrefEmp, which refers to the preferred employee that a client can choose when creating an appointment. This was chosen as intersection data as it is needed for the Appointment entity to know whether the preferred employee is working/free of the service chosen or not, and useful for the Client entity as to show the client what employee they have picked. This intersection data will be inform of the Employee entity's primary key, but with views can be changed later to show the name of an employee since the primary key for that entity is a surrogate key which has no bearing on an actual client.

The cardinalities are as follows:

Appointment – (1,m) – For each Client, and Date & Time, there can be 1, to many appointments which logically makes sense.

Client – (1,1) – For each Appointment, and Date & Time, there can only be 1 client, which also makes sense, since only the client that made the appointment at that time has the appointment reservation, and not any other client.

Edgar Carbajal
CSCI 466-2
Due: 3/1/2023

Date & Time – (1,m) – For each Appointment, and Client, there can be 1 to many Dates and Times, since a client can reserve the same appointment (i.e., with the same services, prefered employee, etc…) at many different dates and times.

*
- reminds
A relationship that links Appointment with Client, in which reminders of a certain appointment is upcoming with a Message(attribute) as intersection data so that the Appointment side can write into, and the Client side can read said message.

The cardinalities are as follows:

Appointment – (1,m) – For each client, there are 1 to many Appointments that the client will be reminded by. This makes sense since a particular client can reserve multiple appointments, thus, they can also receive multiple reminders.

Client – (1,1) – For each Appointment, there are only 1 client they send reminders to. This makes sense since the appointment was reserved by the Client to be used for them only.

*
- bills
A relationship that links Appointment with Client, in which it sends a bill to the client, with intersection data of TaxRate. This intersection data is included in order to calculate the total amount due in which we can find out from the sub-total amount by adding prices from Service instance data through the connection that Service and Appointment have. The total amount can be displayed as a view later in SQL.

The cardinalities are as follows:

Appointment – (1,m) – For each client, there are 1 to many Appointments that the client will be billed for. This makes sense since a particular client can reserve multiple appointments, thus, they can also receive multiple bills (if they show up of course).

Client – (1,1) – For each Appointment, there are only 1 client they send bills to. This makes sense since the appointment was reserved by the Client to be used for them only, so when they show up for their appointment, after they are done, they will be appropriately be billed.

*
- includes
A relationship that links Appointment and Services, in which it represents what services will be done in this particular appointment.

Edgar Carbajal
CSCI 466-2
Due: 3/1/2023

The cardinalities are as follows:

Appointment – (1,m) – For each Service included, there are 1 to many Appointments . This makes sense since a service is not limited to being used on just one Appointment.

Services – (1,m) – For each appointment, there are 1 to many Services included. This makes sense since a client would have freedom in choosing how many services they would like to have done in their appointment.

*
- works
A relationship that links Employee, Service, and Shift together, in which it represents an employee's work schedule. The employee works 1 to multiple shifts in which they can work 1 to multiple services in the Spa. There is intersection data: AmtHours in order to keep track of how many hours are worked in a schedule, for many reasons that an employer would need such as paying employee's accordingly (like overtime), and to follow labor laws, etc.

The cardinalities are as follows:

Employee – (1,m) – For each Shift, and Service worked, there are 1 to many employees working the same shift and service which makes sense.

Service – (1,m) – For each Employee and Shift, there is 1 to many Services being worked which makes sense.

Shift – (1,m) – For each Employee and Service, there are 1 to many shifts being worked, meaning that an employee working a service can be done in many different times and dates which makes sense.

Transforming ER Diagram to Relational DB

Now that the ER Diagram has been properly explained, now we will be transforming it into a Relational DB format in this second section of this PDF.

To transform an ER Diagram to Relational DB in 3NF, we must follow the two main steps, along side their substeps in order.

1.) Handle Entities
- **1a.)** Strong Entities (Not subtypes)
- **1b.)** Strong Entities (Subtypes)
- **1c.)** Weak Entities

2.) Handle Relationships
- **2a.)** Binary Relationships
- **2b.)** Relationships Greater than Binary
- **2c.)** Recursive Relationships

Edgar Carbajal
CSCI 466-2
Due: 3/1/2023

Step 1a.)
For strong entities, all we need to do is make relations/tables of the entities, with their attributes/columns included. Remember we exclude any Date related entities. This includes the Shift entity which is just a Date entity but in a different name. The attributes of these entities will be included later for the tables of relationships.

Transforming our ER Diagram's strong entities we obtain:

Appointment(AppID)
Service(Type, Price)
Client(ClientID, FName, LName, Address, Email, Phone)
Employee(EmpID, FName, LName, Address, Email, Phone, JobTitle, JobSpec, PayRate)

where primary keys are shown with an underline in the attribute/column in the table, and foreign keys are shown with a '*' appended at the end of the column name.

Step 1b.) & Step 1c.)
We have no subtype entities nor weak entities so we are skipping these steps. This makes us finish the first major step in converting the ER Diagram so we move on to the next major step.

Step 2a.)
For binary relationships, we need to split these up into their cardinalities in order to tackle each one effectively.

Binary 1-1:
There are none so we skip this step.

Binary 1-m:
We have two relationships that fit this description in our ER Diagram: "reminds", and "bills". In binary 1-m, we do not have to make a new relation. All we need to do is place the foreign key(the primary key with the 1 side) in the many side. In this case, both relationships are referencing the same two entities, so all we need to do is place the foreign key once. These relationships also have intersection data, which will just be added to the table where the foreign key is placed.

Transforming these two relationships, we obtain:

Appointment(AppID, ClientID*, message, TaxRate)

where the now updated list of tables along side the foreign key list is below:

Service(Type, Price)
Appointment(AppID, ClientID*, Message, TaxRate)
Client(ClientID, FName, LName, Address, Email, Phone)
Employee(EmpID, FName, LName Address, Email, Phone, JobTitle, JobSpec, PayRate)

Edgar Carbajal
CSCI 466-2
Due: 3/1/2023

List of foreign keys:

ClientID – Home Table: **Client**

where primary keys are shown with an underline in the attribute/column in the table, and foreign keys are shown with a '*' appended at the end of the column name.

Binary m-m:
We have one binary m-m relationship which is: "includes". For these, we must make a new table, and include both primary keys of the entities as foreign keys and make them the new primary keys of the new table.

Converting this relationship, we end up with the following:

Includes(AppID*, Type*)

where the now updated list of tables along side the foreign key list is below:

Service(Type, Price)
Includes(AppID*, Type*)
Appointment(AppID, ClientID*, Message, TaxRate)
Client(ClientID, FName, LName, Address, Email, Phone)
Employee(EmpID, FName, LName, Address, Email, Phone, JobTitle, JobSpec, PayRate)

List of foreign keys:

ClientID – Home Table: **Client**
AppID – Home Table: **Appointment**
Type – Home Table: **Service**

where primary keys are shown with an underline in the attribute/column in the table, and foreign keys are shown with a '*' appended at the end of the column name.

Step 2b.)
Next is to check any relationships greater than binary. In these relationships, we must always create a new table, in which it includes all of the primary keys of the entities connected as foreign keys, and chooses the primary keys out of these foreign keys by checking which foreign key is connected to an entity that is on a "many" side like when doing it in binary relationships.

In any case, this is easier to see which will be primary keys in the new table by making functional dependency lines lines which will be done to easily see how to convert these relationships.

Edgar Carbajal
CSCI 466-2
Due: 3/1/2023

We have two relationships that are greater than binary which are: “reserves” and “works”. Let us start with “reserves” first.

For “reserves”, we have two “many” sides, and one “1” side in which the functional dependency would look like so:

$$\text{AppID, Datetime} \rightarrow \text{ClientID, PrefEmp}$$

so we can infer from this that the left-hand side should become the new primary keys of the new relation, and all of them will become foreign keys normally; however, since Datetime is from an entity that is actually a Date-like entity, it will just be a regular attribute, and not a foreign key, since there is no relation of the entity it originates from. Remember, PrefEmp is intersection data so it will just be treated as a regular attribute in the new table.

Transforming this relationship, we obtain:

Reserves(AppID*, Datetime, ClientID*, PrefEmp)

Now for “works”, we have all sides being “many”, so that means we have no functional dependencies, thus all primary keys of the entities that will become foreign keys, must also become the primary key together for the new table. Again, this new table also has primary keys from a Date-like entity, so they will not be foreign keys, and also has intersection data which will be dealt with as a regular attribute.

Transforming this relationship, we obtain:

Works(EmpID*, Type*, Date, StartTime, EndTime, AmtHours)

Now having done all relationships greater than binary we can move on, but first an update on what tables we have now obtained:

Service(Type, Price)
Includes(AppID*, Type*)
Appointment(AppID, ClientID*, Message, TaxRate)
Client(ClientID, FName, LName, Address, Email, Phone)
Reserves(AppID*, Datetime, ClientID*, PrefEmp)
Works(EmpID*, Type*, Date, StartTime, EndTime, AmtHours)
Employee(EmpID, FName, LName, Address, Email, Phone, JobTitle, JobSpec, PayRate)

Edgar Carbajal
CSCI 466-2
Due: 3/1/2023

List of foreign keys:

ClientID – Home Table: **Client**
AppID – Home Table: **Appointment**
Type – Home Table: **Service**
EmpID – Home Table: **Employee**

where primary keys are shown with an underline in the attribute/column in the table, and foreign keys are shown with a '*' appended at the end of the column name.

Step 2c.)
There are no recursive relationships in this diagram, so we skip this step.

****
We are now done transforming the ER Diagram into a Relational DB. The final list will be printed again below. This will be the basis on how the SQL queries/code will be written to create the exact Relational DB you see below in MariaDB. The code/queries will be written in a separate text file.

Final List of Tables/Relations:

Service(Type, Price)
Includes(AppID*, Type*)
Appointment(AppID, ClientID*, Message, TaxRate)
Client(ClientID, FName, LName, Address, Email, Phone)
Reserves(AppID*, Datetime, ClientID*, PrefEmp)
Works(EmpID*, Type*, Date, StartTime, EndTime, AmtHours)
Employee(EmpID, FName, LName, Address, Email, Phone, JobTitle, JobSpec, PayRate)

Final List of Foreign Keys:

ClientID – Home Table: **Client**
AppID – Home Table: **Appointment**
Type – Home Table: **Service**
EmpID – Home Table: **Employee**

where primary keys are shown with an underline in the attribute/column in the table, and foreign keys are shown with a '*' appended at the end of the column name.