# Database Requirement

## Scenario

You need to present an overview of your database system.

I’m Saif Haddad, I’m an IT specialist at Microsoft company, I was approached by Al Basheer Hospital to build a database for their Department of Ophthalmology.

The Hospital asked for a system that could store and manage patient information, ophthalmologist information, eye examination information, treatment information, and emergence contact information, this information was collected from the management team and the ophthalmology department.

Another meeting was held with the hospital management team and the ophthalmology department so they can explain to me what information they need to store about the patient, ophthalmologist, eye examination, treatment, emergence contact .

And the result of this meeting was that they need to know and store some personal information about the patient such as the patient name, phone number and date of birth, and some other information like the medical history and the insurance name and the insurance degree and login information (username, password).

Also, they need to know and store some personal information about the ophthalmologist such as the ophthalmologist’s name and his phone number in addition to some other information like the specialty and his medical degree and his work schedule and login information (username, password).

Also, the need to know and store some information about the eye examination such as the date of the examination and the type of the examination and the results of the examination.

Also, they need to know and story some information about the treatment such as the treatment type and the dosage and frequency of the treatment and when is the following up appointment.

And the last request they asked for is that the also need to know and story some information about the emergence contact such as the name and the relationship and the phone number of the required person.

After agreeing on all parts of the project, on the cost, payments, and the period provided, the project was accepted.

## Data Requirements

1- Each patient has a unique ID, Name, Date of birth, one phone number, Insurance details, and multiple medical history, username, password.

2- Each ophthalmologist has a unique ID, Name, specialty, multiple medical degree, One phone number, and work schedule, username, password.

3- Each eye examination has a unique ID, date, type of examination, and results of examination

4- Each treatment has a unique ID, type of treatment, dosage and frequency of medication, and follow-up appointments.

5- Each emergence contact has a unique ID, name, relationship to the patient, and contact information.

1-Patient:

- A patient can have multiple eye examinations.

- A patient can have multiple treatments.

- A patient can have one emergence contact.

-A patient can see multiple ophthalmologists.

2-Ophthalmologist:

- An ophthalmologist can perform multiple eye examinations

- An ophthalmologist can prescribe multiple treatments.

- An ophthalmologist can be seen by many patients.

3- Eye examination:

- An eye examination is associated only with one patient.

- An eye examination is associated with only one ophthalmologist.

4-Trearment:

- A treatment is associated with only one patient.

- A treatment is associated with only one ophthalmologist.

- A treatment is associated with only one eye examination.

5-Emergence call:

-A emergence call belongs only for one patient.

A patient can have one emergence contacts, and each emergence contact is associated with only one patient.

A patient can have multiple eye examinations, but each eye examination is associated with only one patient.

A patient can visit multiple ophthalmologists, and each ophthalmologist can see multiple patients.

A patient can receive multiple treatments, but each treatment is associated with only one patient.

An ophthalmologist can perform multiple eye examinations, but each eye examination is associated with only one ophthalmologist.

An ophthalmologist can prescribe multiple treatments, but each treatment is associated with only one ophthalmologist.

## User and System Requirement

You need to present the users of the database by listing the users’ groups. Then you are required to mention the way that each user will interact with the system by defining their needs.

**Note:** Think of this part as displaying the users and functionalities of the user interface.

1-patient:

- View follow-up appointments. (Treatment)

- Update and edit personal information (contact information (Patient), insurance details, emergency contact).

- View medical history.

- View treatment and medication information.

- Receive reminders and notifications about follow-up appointments.

2-Ophthalmologist:

- View work schedule.

- View the patient's personal information such as name, date of birth, contact information,

and insurance details.

- Diagnosing and evaluate the patient's condition

-Write a report after finishing the Eye examination based on his Diagnoses and evaluation.

-Prescribe treatments for patient based on the eye examination result.

- View results of previous eye examinations.

- View previous treatments received for patients.

- View the patient's medical history.

# Database Design

## Conceptual Design

**What is conceptual design:**

Is the first stage of the design process, during which the designer investigates several concepts and conceptualizes the general design strategy. Also, Establishing the project's fundamental needs and creating a preliminary design plan based upon the previously identified data requirements. This phase or this design will contain the entities that should exist and the connections (relationships) between the entities. ( Conceptual, 2019)

**Note:** Sometimes the conceptual design is called an entity relationship design because the main prepose of this design is to show the entities and the relationships.

**The goals of the conceptual design:** ( Conceptual, 2019)

- Clearly define the project's direction.

- Create a preliminary design plan that complies with the project requirement.

- provide a precise description of the finished service's appearance and functionality.

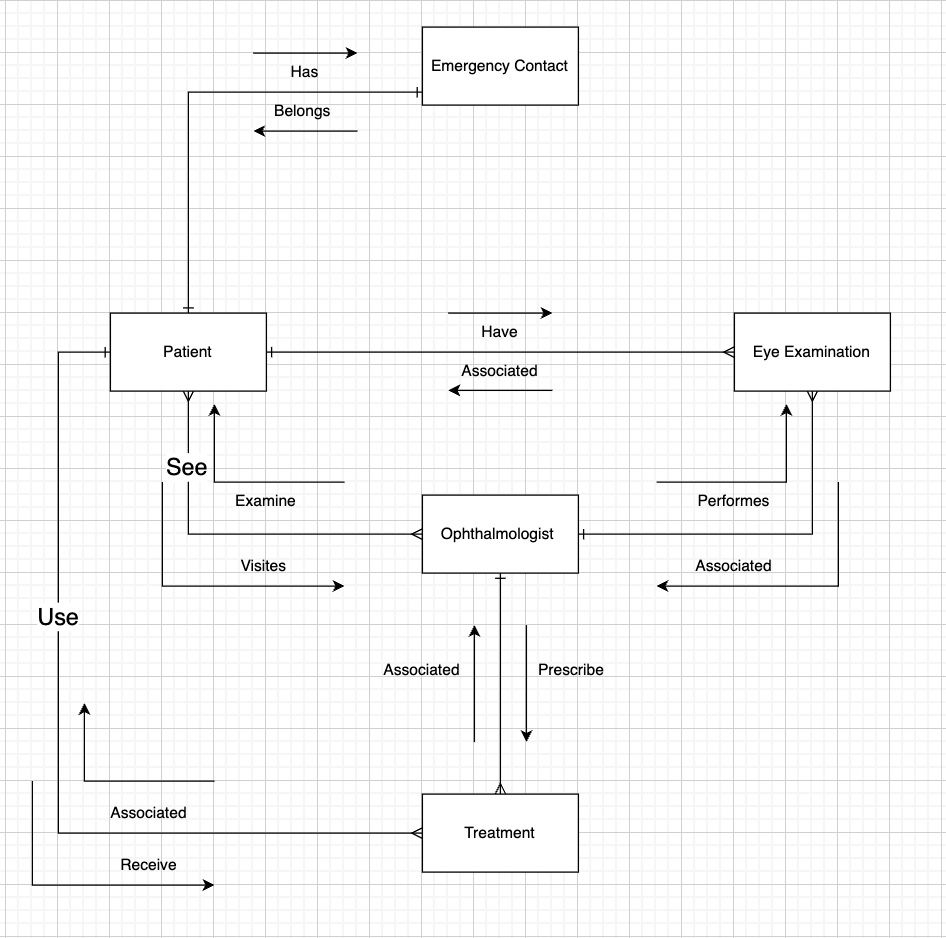
- result in a design solution that is more effective.

**The components:**

Entities

Relationships

**My conceptual design:**



This is my conceptual design based on the written scenario, as we can see I have five entities: Patient, Ophthalmologist, Eye examination, Treatment, Emergency contact.

Every entity is linked with one or more entities by a lines and these seen lines are called relationships, and in my conceptual design I have the three types of relations ships which are one to one relationship, one to many relationship and many to many relationship.



**Note:** In every place in the above picture, we see this sign on the relationship lines it means that it is one relationship.

In every place we see this sign on the relationship lines it means that it is Many relationship (this sign called the Crow’s feet)

**Starting with the Patient entity:**

As we can see in the picture that the Patient entity is linked with four other entities (Emergency contact, Eye examination, treatment, and Ophthalmologist):

It is linked by one to one relationship with the Emergency contact entity because A patient can have one emergence contacts, and each emergence contact is associated with only one patient, also it is linked by one to many relationship with the Eye examination entity because A patient can have multiple eye examinations, but each eye examination is associated with only one patient, also it is linked by one to many relationship with the Treatment entity because A patient can receive multiple treatments, but each treatment is associated with only one patient, and the last entity that it is linked with by a many to many relationship is the Ophthalmologist entity because A patient can visit multiple ophthalmologists, and each ophthalmologist can see multiple patients..

One to One relationship: Patient (One) – Emergency contact (One).

One to Many relationship: Patient (One) – Eye examination (Many).

One to Many relationship: Patient (One) – Treatment (Many).

Many to Many relationship: Patient (Many) – Ophthalmologist (Many).

**Moving on to the Ophthalmologist entity:**

As we can see in the picture that the Ophthalmologist entity is linked with three other entities (Eye examination, Treatment and Patient):

It is linked by one to many relationship with the Eye examination entity because An ophthalmologist can perform multiple eye examinations, but each eye examination is associated with only one ophthalmologist, also it is linked by one to many relationship with the Treatment entity because An ophthalmologist can prescribe multiple treatments, but each treatment is associated with only one ophthalmologist, and the last entity that it is linked with by a many to many relationship is the patient entity because A patient can visit multiple ophthalmologists, and each ophthalmologist can see multiple patients.

One to Many relationship: Ophthalmologist (One) – Eye examination (Many).

One to Many relationship: Ophthalmologist (One) – Treatment (Many).

Many to Many relationship: Ophthalmologist (Many) – Patient (Many).

**Moving on to the Eye examination entity:**

As we can see in the picture that the Eye examination entity is linked with three other entities (Ophthalmologist, Treatment and Patient):

It is linked by one to many relationship with the Patient entity because A patient can have multiple eye examinations, but each eye examination is associated with only one patient, and the last entity that it is linked with by one to many relationship is the Ophthalmologist entity because An ophthalmologist can perform multiple eye examinations, but each eye examination is associated with only one ophthalmologist.

One to Many relationship: Eye examination (Many) – Patient (One).

One to Many relationship: Eye examination (Many) – Ophthalmologist (One).

**Moving on to Treatment entity:**

As we can see in the picture that the Treatment entity is linked with three other entities (Eye examination, Ophthalmologist and Patient):

It is linked by one to many relationship with the Patient entity because A patient can receive multiple treatments, but each treatment is associated with only one patient, and the last entity that it is linked with by one to many relationship is the Ophthalmologist entity because An ophthalmologist can prescribe multiple treatments, but each treatment is associated with only one ophthalmologist.

One to Many relationship: Treatment (Many) – Patient (One).

One to Many relationship: Treatment (Many) – Ophthalmologist (One).

**Moving on to the last entity which is the Emergency contact:**

As we can see in the picture that the Emergency contact entity is linked just with one entity (Patient):

It is linked by one to one relationship with the Patient entity.

One to One relationship: Emergency contact (One) – Patient (One).

## Schema and Mapping

**Before**

-Patient table (Patient\_ Id, Insurance\_Id, Name, Phone number, Date of Birth, Medical history, Username, Password, Insurance degree, Insurance name).

-Ophthalmologist table (Ophthalmologist\_Id, Name, Specialty, Phone number, Medical degree, Work schedule, Days, Username, Password).

-Eye examination table (Eye examination\_Id, Patient\_Id, Ophthalmologist\_Id, Date, Type of examination, Results of examination).

-Treatment (Treatment\_Id, Patient\_Id, Ophthalmologist\_Id, Type of -treatment, Follow-up appointments, Dosage, Frequency).

-Emergency Contact table (Contact\_Id, Patient\_Id, Name, Relationship, Phone number)

**After:**

-Patient (Patient\_ Id, Insurance\_Id First name, last name, Phone number, Date of Birth, Username, Password, insurance degree)

-Insurance (Insurance\_Id, Insurance name)

-Patient\_Medical history (Patient\_Id, Medical history)

-Ophthalmologist table (Ophthalmologist\_Id, First name, last name, Phone number, Specialty, Username, Password)

-Work schedule (Ophthalmologist\_Id, Day, Work schedule)

-Ophthalmologist\_Medical degree (Ophthalmologist\_Id, Medical degree).

-Eye examination table (Eye examination\_id, Patient\_Id, Ophthalmologist\_Id, examination Date, Type of examination, Results of examination)

-Treatment (Treatment\_Id, Patient\_Id, Ophthalmologist\_Id, Dosage, Frequency, Follow-up appointments, Treatment type)

-Emergency Contact table (Contact\_Id, Patient\_Id, First name, Last name, Relationship, Phone number)

-Ophthalmologist-Patient table (Ophthalmologist\_Id, Patient\_Id)

## Normalization

### 1st NF

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | Attributes | Violation description | Solution – Relations |
| The relations schema | The attribute name | Describe why it is not in the 1st NF (the violation) | Show the schema for each affected relation. |
| Patient | Name | Because the Name attribute which is in the Patient entity is not atomic, it is composed of two sub attributes (First name, Last name, which means it is composite attribute | Just splitting the attribute Name in to (Fist name, Last name)  Patient (Patient\_Id, First name, last name, Phone number, Date of Birth, company Insurance\_Id, Insurance name, Insurance degree, Username, Password) |
| Patient | Medical history | The "Medical history" attribute is multi-valued, and it must be atomic value  If we leave it in the same table, we will have a lot of nulls values | Making a new table that contains the patient Id and the Medical degree attribute  Patient\_Medical history (Patient\_Id, Medical history) |
| Ophthalmologist | Name | Because the Name attribute which is in the Ophthalmologist entity is not atomic, it is composed of two sub attributes (First name, Last name, which means it is composite attribute | The attribute Name (Fist name, Last name)  Ophthalmologist table (Ophthalmologist\_Id, First name, Mid name, last name, Phone number, Specialty, Work schedule) |
| Ophthalmologist | Medical degree | The Medical degree attribute is multi-valued, and it must be atomic value  If we leave it in the same table, we will have a lot of nulls values | Making a new table that contains Ophthalmologist\_Id and the medical degree  Ophthalmologist\_Medical degree (Ophthalmologist\_Id, Medical degree). |
| Emergence contact | Name | Because the Name attribute which is in the Ophthalmologist entity is not atomic, it is composed of two sub attributes (First name, Last name, which means it is composite attribute | Just splitting the attribute Name to (Fist name, Last name)  Emergency Contact table (Contact\_Id, Patient\_Id, First name, Last name, Relationship, Phone number) |

Each column in the table should have just atomic values (Single value), which means that multi values and repeatable values are not acceptable.

If we leave the composite attributes and we don’t separate them into sub attributes, will lead to a lot of problems:

1. Data redundancy: Usually composite attributes have repeated values, Leaving them as one property may result in duplication in the data, which may raise storage needs and make consistency more difficult to maintain.
2. Data manipulation difficulties: Composite attributes, particularly those that contain many pieces of information, can be challenging to work with. In this case, processing the composite "Name" component to get the last name would be necessary to search for all ophthalmologists with the last name "Smith." This process can be time-consuming

Overall, keeping composite attributes in a database might make it harder to manage data, the need more storage, and affect performance. For both data integrity and usefulness, it is therefore best to normalize the table and divide composite attributes into their atomic components.

### 2nd NF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relations | FDs | Violation description | | Solution – Relations |
| The relations schema | Show the functional dependencies causing the violation | Describe why it is not in the 2nd NF (the violation) | Show the schema for each affected relation. | |
| Ophthalmologist | The Work schedule attribute depends on the Day attribute primary key, but the work Schedule attribute does not depend on the main primary key which is Ophthalmologist\_id  Ophthalmologist\_Id, Day > Work schedule  But the rest of the attributes which are First name, last name, Phone number, Specialty, Username, Password depends on the Ophthalmologist\_Id primary key but they don’t depend on the Day attribute primary key  Ophthalmologist\_Id} > First name, last name, Phone number, Specialty, Username, Password | There is a partial dependency between the Work schedule attribute and Day attribute primary key, this means it is only dependent on the day attribute primary key not on all the composite primary key this is the violation. | Ophthalmologist table (Ophthalmologist\_Id, day, Name, Specialty, Phone number, Medical degree, Work schedule, Username, Password).  We can see that the attribute work schedule was removed and also the primary key day and was taken to a new table  (Ophthalmologist\_Id, Day, Work schedule)  This is the new table and this how we solve the partial dependency relationship | |

if we leave the partial dependence in the same table will cause a lot of problems:

1. Data redundancy and inconsistencies: Partial dependencies may result in duplicate information being saved in a table, causing the data to be inconsistent and inaccurate. As a result, maintaining data integrity may be challenging, and queries or data updates might run into issues.
2. Update anomalies: Update anomalies can also be caused by partial dependencies, when changing one attribute in a record requires modifying other values in the same record. This may make it challenging to keep data consistency and may result in data inaccuracies.
3. Poor performance: Partial dependencies may cause the database to need to do additional operations to resolve them, which can result in poor performance when searching or editing the data.

### 3rd NF

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | FDs | Violation description | Solution – Relations |
| The relations schema | Show the functional dependencies causing the violation | Describe why it is not in the 3rd NF (the violation) | Show the schema for each affected relation. |
| Patient | The insurance name attribute is dependent on the Insurance\_Id attribute  Insurance\_Id >Insurance name  And the Insurance\_Id depends on the Patient\_Id primary key attribute  But the rest of the attributes which are Insurance\_Id, Name, Phone number, Date of Birth, Medical history, Username, Password are dependent on the Patient\_Id  Patient\_Id > Insurance\_Id, Name, Phone number, Date of Birth, Medical history, Username, Password | There are a transitive dependency between the Patient\_Id primary key attribute and the Insurance name attribute, as the Insurance name is dependent on the Insurance\_Id  This is the violation | Patient (Patient\_ Id, First name, last name, Phone number, Date of Birth, Username, Password, insurance degree)  As we can see that we remove the Insurance\_id attribute from the Patient table and also we removed the Insurance name attribute from the Patient table  New entity:  Insurance (Insurance\_Id, Insurance name)  This is the new table I created to solve the transitive dependency |

if we leave the transitive dependence in the same table will cause a lot of problems:

1. Data redundancy: For every patient with the same insurance, the same insurance name will be used again, resulting in duplicate data that might be challenging to maintain and update.
2. Inconsistencies: There will be inconsistencies in the data if a specific insurance company's name is changed, as this would need updating all of the patients who have that insurance.
3. Insertion anomalies: In order to avoid insertion anomalies, we would need to insert a row into the Patient database with null values for the insurance name and degree if a new insurance provider was added to the system but no patients had that insurance yet.

## Logical Design

**What is logical design:** (Inc, 2023)

It is a phase in data base design, it came after the conceptual design also it depends so much on the conceptual design in what entities there are and in how the entities are related to each other, and it consider as a step to convert the conceptual data into logical data design that can be implemented in the design

**The goals of the logical design:** (Inc, 2023)

- Accurately represent the needs of the business: All important data entities, attributes, and

relationships needed that faces the needs of the business.

- To normalize the data model and get clear of redundant and inconsistent data.

- To build specific relationships between entities and specify their attributes

- Must be certain that the database is flexible to future modifications and upgrades and is

scalable.

- By using constraints, it is possible to ensure data consistency and integrity.

**By these points we can ensure the data integrity:** (Inc, 2023)

- A primary key constraint: makes sure that a certain column or a set of columns

uniquely identify each row in a table.

- A foreign key constraint: makes sure that the values of two columns in unique

tables match.

-Not null constraint: makes sure a column can't have any null values.

- Unique constraint: makes sure that every value in a column is unique.

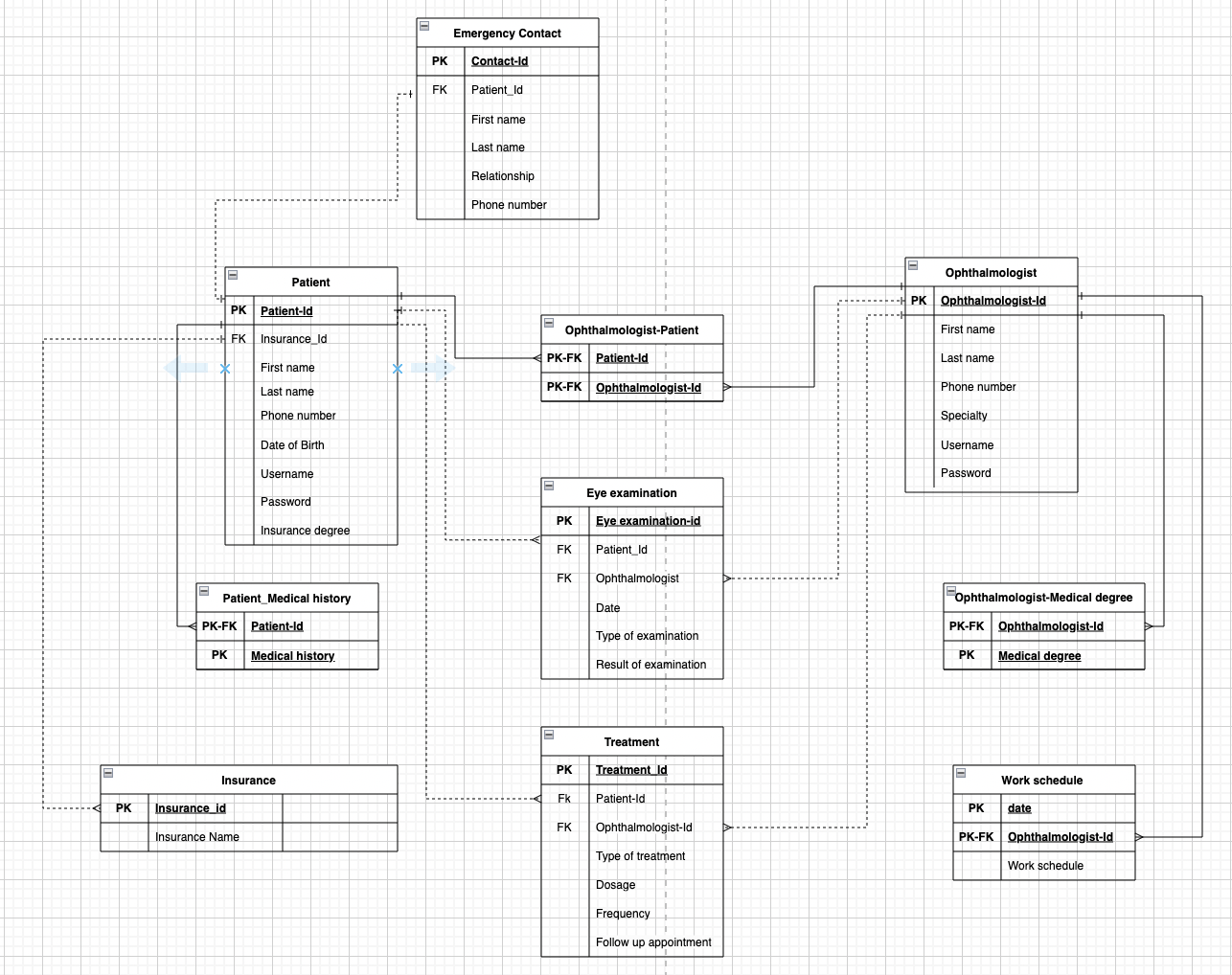
**The components of logical design:** (Inc, 2023)

-Entities: it is the Tables

-Attributes: the key types (Primary key, Foreign key)

-Relationships: we put the relationships(one to one, many to many, one to many) and also we specify if it is strong relationship or weak relationship

**This is my logical design:**



As you can see I have ten tables or entities which are (Patient, Ophthalmologist, Treatment, Eye examination, Patient-Medical history, Ophthalmologist-medical degree, Work schedule, Ophthalmologist-Patient, Insurance, Emergence contact).

these tables contains attributes and the primary keys and the foreign keys, also these tables are linked with each other’s based on relationships

I have some strong relationships and I have some weak relationships:

Strong relationship: The connected data must be strongly related to and dependent upon one another in order for there to be a strong relationship. This implies that the data in one entity depends on the data in the other entity in order to exist.

Weak relationship: where the related variables are less dependent on one another. Accordingly, data in one entity can exist independently of data in another entity.

**The attribute of each table:**

The Patient table contains information about each patient: Patient\_Id, Insurance\_Id, First name, Last name, Phone number, Date of birth, Username, Password.

The Ophthalmologist table stores information about each ophthalmologist: Ophthalmologist\_Id, First name, last name, Phone number, Specialty, Username, Password.

The Emergence contact table keeps track of each patient's emergency contact information.: Contact-Id, Patient\_Id, First name, last name, Relationship, Phone number.

The Ophthalmologist-Patient table represents the relationship between ophthalmologists and their patients: Ophthalmologist\_Id, Patient\_Id.

The Patient-Medical history table keeps track of the medical history of each patient: Patient\_Id, Medical history.

The Insurance table contains information about the different insurance providers: Insurance\_Id, Insurance name.

The Ophthalmologist-Medical degree table stores the medical degree of each ophthalmologist: Ophthalmologist\_Id, Medical degree.

The Work schedule table tracks the work schedule of each ophthalmologist: Ophthalmologist\_Id, day, Work schedule.

The treatment table stores information about the treatment of each patient: Treatment\_Id, Patient\_Id, Ophthalmologist\_Id, Type of treatment, Dosage, Frequency, Follow up appointment.

The Eye examination table keeps track of each eye examination: Eye examination\_Id, Patient\_Id, Ophthalmologist\_Id, Date, Type of examination, Results of examination.

The patient table: it has the Patient\_Id as w primary key and the Insurance\_Id as foreign key.

The Patient\_Medical history Table: it has the Patient\_Id as primary and foreign key at the same time and the Medical history as Primary key.

The insurance table: it has the Insurans\_Id as a primary key.

The Ophthalmologist table: it has the Ophthalmologist\_Id as a primary key.

The work schedule table: it has the Ophthalmologist\_Id as primary and foreign key at the sametime and the day as primary key.

The Eye examination table: it has the Eye examination\_Id as primary key and the Ophthalmologist\_Id and the Patient\_Id as foreign key.

The treatment table: it has the treatment\_Id as primary key and the Ophthalmologist\_Id and Patient\_id as foreign key.

The Emergency Contact table: it has the Contact\_id as primary key and the Patient\_Id as foreign key.

the Ophthalmologist-Patient table: it has the Ophthalmologist\_Id as foreign key and primary key at the sametime and also the Patient\_Id as foreign key and primary key at the sametime

**The relationships:**

Patient and Emergence contact are linked by one to one relationship, and it is weak relationship

Patient and Patient\_Medical history are linked by one to may relationship and it is strong relationship.

Patient and Ophthalmologist-Patient are linked with one to many relationship and it is a strong relationship

Patient and Eye examination are linked with one to many relationship and it is weak relationship

Patient and treatment are linked with one to many relationship and it is a weak relationship

Insurance and Patient are linked by one to many relationship and it is a weak relationship

Ophthalmologist and Ophthalmologist-Patient are linked by one to many relationship and it is a strong relationship.

Ophthalmologist and Ophthalmologist-Medical degree are linked by one to many relationship and it is strong relationship.

Ophthalmologist and Work schedule are linked by one to many relationship and it is strong relationship.

Ophthalmologist and Eye examination are linked by one to many relationship and it is a weak relationship.

Ophthalmologist and treatment are linked by one to many relationship and it is weak relationship.

## Physical Design

**What is physical design:** (Rick BanisterRick is the Founder and CEO of Sesame Software. , 2022)

It is a phase in data base design, it came after the logical design, in this phase we converting a database's logical design into a physical representation that may be used with a certain database management system

**The goals of the physical design:** (Rick BanisterRick is the Founder and CEO of Sesame Software. , 2022)

the main goal is to shows all entity structures as well as the primary key, foreign key, attribute restrictions, attribute data type, and relationships between tables.

1. Optimize performance: The physical design seeks to enhance database performance by reducing the amount of time needed to access and alter data. This entails selecting the most effective indexing and partitioning techniques, as well as storage and retrieval technologies.
2. Ensure scalability: The physical design need to be scalable, able to manage an expanding number of users and data without decreasing in performance.
3. Ensure availability: The database should be available, which means that users and programs may access it when needed, thanks to the physical design.

**The components:** (Rick BanisterRick is the Founder and CEO of Sesame Software. , 2022)

1-Entities

2-attributes

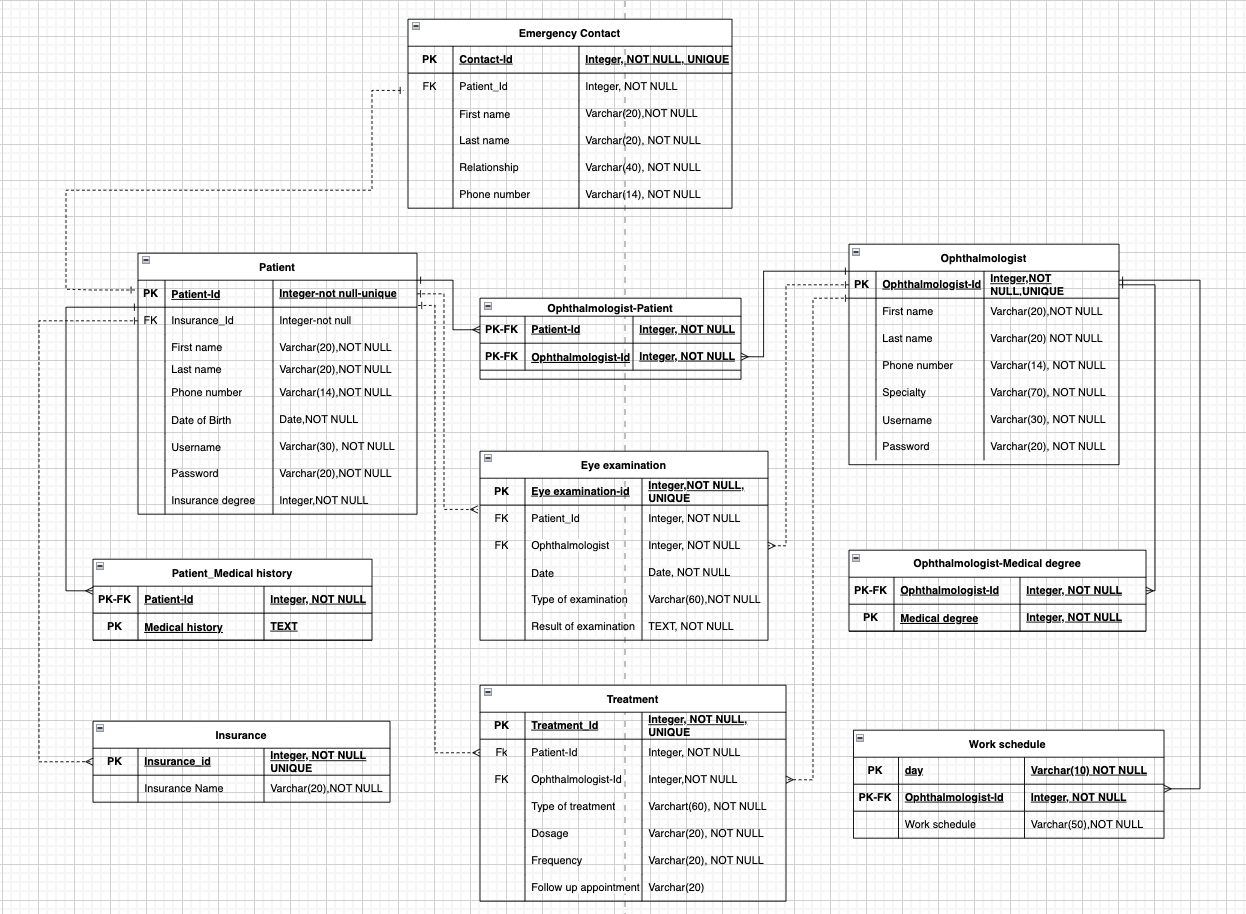
3-Primary keys

4-Foreign keys

5-attributes data types

6-relations between the entites

**This is my physical design:**

****

Regarding the physical design that I did, I ended up taking the logical design that was done earlier and added the needed data types for each attribute. Within my physical design I ended up using the UNQIUE, NOT NULL, TEXT, INTEGER, VARCHAR, and DATE datatypes. Each of which has their own reason onto why they should be used.

• The UNQUE data type is used since it only defines within a row that there is a unique value onto that column.

• The NOT NULL data type is used to specify that the attribute with this given data type must have a value and must not be NULL.

• The TEXT data type is used for the attributes that are in need of a longer text field.

• The INTEGER data type is used since it only stores integer values onto the attribute that has it.

• The VARCHAR data type is used to help store variables according to the amount that it is set to store.

• The DATE data type is used to store only date values.

## Effectiveness of the design

Based on what I have done from the beginning, I have choses the entities, and these entities was chosen based on the information that provided by the management team and the department of ophthalmologist and also on the requirements.

The Patient entity was chosen based on the needs of the hospital, because they need to know and store some personal information about the patient such as the patient name, phone number and date of birth, and some other information like the medical history and the insurance name and the insurance degree, also extra login details for the patient to view their data which are the username and password

The Ophthalmologist entity was chosen based on the needs of the hospital, because they need to know who work at the hospital and store some personal information about the ophthalmologist such as the ophthalmologist’s name and his phone number in addition to some other information like the specialty and his medical degree and his work schedule and login information (username, password).

The Eye examination entity was chosen based on the needs of the hospital, because they need to know and store some information about each eye examination that is performed at the hospital such as the date of the examination and the type of the examination and the results of the examination

The Treatment entity was chosen based on the needs of the hospital, because they need to know and story some information about the treatments that patients receive at the hospital such as the treatment type and the dosage and frequency of the treatment and when is the following up appointment.

The Emergence contact entity was chosen based on the needs of the hospital, because the need to know and story some information about the people listed as each patient's emergency contacts such as the name and the relationship and the phone number of the required person.

Moving on to the relationships, they were chosen based on the data nature and on the requirement of the hospital and based on the logic of real life:

Starting with the relationship which is between the Patient table and the Emergence contact table:

Due to the fact that each patient can only have one emergency contact and that each emergency contact can only be connected with one patient, the relationship between the "Patient" and "Emergency Contact" entities is one-to-one. For example, if a patient names their spouse as their emergency contact, just that patient would be connected to that emergency contact. In this scenario, each record's "Patient\_Id" in the "Emergency Contact" entity would be unique, suggesting that it is only linked to a single patient.

Moving on to the relation between the Patient and the eye examination:

One patient can get several eye exams, but only one patient is connected with each eye examination, making the relationship between the "Patient" entity and the "Eye examination" entity one-to-many. The outcomes of each examination would be entered in a separate record in the "Eye examination" entity, for for example, if a patient had an eye exam every six months. In this scenario, each record connected to the patient would have the same "Patient\_Id" in the "Eye examination" object.

Moving on to the relation between the ophthalmologist and the eye examination:

One ophthalmologist can do several eye examinations, but only one ophthalmologist is connected with each eye examination, so the link between the "Ophthalmologist" entity and the "Eye examination" entity is one-to-many. The results of each eye examination would be documented in a distinct record in the "Eye examination" entity, as an ophthalmologist may perform many eye examinations on multiple patients each day. In this scenario, all entries connected to that ophthalmologist would have the same "Ophthalmologist\_Id" in the "Eye examination" entity.

Moving on to the relation between Patient and Treatment:

One patient may get several treatments, but only one patient is connected with each treatment, so the relationship between the "Patient" entity and the "Treatment" entity is one-to-many. For example, after receiving eye drops, a patient can be given a follow-up visit two weeks later. The "Patient\_Id" in the "Treatment" entity would be the same for all records connected to that patient, and every detail of each treatment would be documented in an own record in the "Treatment" entity.

Moving on to the first normal form which solve the composite attributes and the multi values attribute:

I consider the name attribute in the patient entity a composite attribute because we can divide the attribute in to sub attributes

It is preferable to divide the composite property "Name" into sub attributes for the first name and last name in the Patient database. Patients may now be found and sorted more quickly by either their first or last name. When the whole name is given, it also removes any uncertainty over whether the first or last name is being used.

I consider the Medical history attribute in the Patient table is a multi-value attribute because one Patient can have more than Medical history:

It is recommended to develop a new entity with a one-to-many link to the Patient table in order to handle the multivalued attribute "Medical history" in the Patient table. This makes it simpler to handle and search for specific patients' medical problems. As an alternative, the patient table may store the medical history as a list or array, but this would make it more difficult to quickly search and evaluate the data.

There is no other effective ways to solve the composite values and the multivalued attributes

Splitting the name attribute and making a new entity for the medical history attribute is the only approach that works as well. For example, the complete name might be saved as a single attribute and then processed as needed, as instead of dividing the name attribute. This strategy, however, would involve more processing steps and risk mistake. The information might be kept in the Patient database as a list separated by commas rather than as a distinct object for the medical history property. This strategy, however, would restrict the data's accessibility for querying and analysis and may lead to errors or data duplication.

Moving on to how I solve the one to one relationship between entities:

For example:

One-to-one relationship is between the "Patient" entity and the "Emergency Contact" entity. This indicates that each emergency contact can only be linked to one patient and that each patient can only have one emergency contact.

Patient\_Id, the main key of the "Patient" entity, is added to the "Emergency Contact" object as a foreign key. Each emergency contact record is connected to a matching patient record by use of a foreign key. The "Emergency Contact" entity additionally defines the foreign key as a unique key, providing that each emergency contact is connected to just one patient.

By applying referential integrity requirements, this technique enables for quick searching of pertinent information between the two entities and protects data integrity. The one-to-one relationship may be lost if there wasn't a specific foreign key in the "Emergency Contact" entity that prevented multiple emergency contacts from being linked to the same patient. The most suitable way to model one-to-one relationships is to add a foreign key to the other entity.

Moving on to how I solve the one to many relationship between entities

Each instance of the "one" entity can be linked to lots of instances of the "many" entity in a one-to-many relationship, but every instance of the "many" entity can only be linked to one instance of the "one" entity. The primary key of the "one" object is added as a foreign key to the "many" entity to express that relationship in database. The instances of the "many" entity are linked to the matching instance of the "one" entity by this foreign key.

For example:

A one-to-many link is available between the "Patient" entity and the "Eye examination" and "Treatment" entities. The "Patient\_Id" primary key of the "Patient" entity is included as a foreign key in each of the "Eye examination" and "Treatment" entities to show these relationships. As a result, we are able to connect every instance of the "many" entity to its matching instance of the "one" object.

This is the most effective way to link the on to many relationships entities

Other options, such putting the relevant data in a single table using normalization techniques, might result in redundant data, inconsistent data, as well as performance problems. In a relational tables, the ideal approach to express one-to-many connections is to take the primary key of the "one" entity and add it as a foreign key to the "many" entity.

Moving on to how I solve the many to many relationship between entities

In this example:

The "Patient" and "Ophthalmologist" entities have a many-to-many relationship in the data provided. We build a new entity  that has foreign keys to both the "Patient" and the "Ophthalmologist" entities in order to describe this relationship. By using the new table entity, we may link various instances of the "Patient" entity to various instances of the "Ophthalmologist" entity.

Other options, such as making a single table that stores all the data, may result in data duplication, consistency problems, and performance problems. In this case, if we anomaly the data by include a column in the "Patient" table to hold the identification numbers of the "Ophthalmologist" entities with which they have appointments, we will need to repeat this information for each appointment. When we need to make changes or cancel appointments, in specific, this might result in a lot of data duplication and inconsistencies.

So this option is not valid

Moving on to the normalization:

The practice of normalizing data in a relational database removes dependencies and duplications, resulting in a more effective and uniform database design. It makes sure that each table retains only one set of relevant data, and that the only thing the primary key identifies for each attribute. This decreases data redundancy, assures data consistency, and makes data maintenance easier, all of which contribute to a more dependable and effective database.

Note: to prof what I said you can see what I have wrote about the multi valued attribute in 2.6.

If we did not apply any form of the normalization will effect a lot, you can found what it effect and how in 2.3.1 and 2.3.2 and 2.3.3

Moving on to the last thing which is to Explain why we made all the decisions of the datatypes and how it affects the effectiveness of the database?

Columns with the NOT NULL constraint assist to guarantee that the data recorded in the column is accurate and relevant. It avoids inserting of NULL values, which could lead to manipulating data. We make sure that important data is always present and accurate by making them not nulls, like Patient\_Id, Insurance\_Id, First name, Last name, Phone number, Date of Birth, Specialty, Username, Password, and Type of examination, NOT NULL.

As a result of the UNIQUE constraint, duplicate entries are avoided by ensuring that each value in a column have a unique value. In order to maintain data integrity, we used UNIQUE constraints for different columns in our design, including Patient\_Id, Insurance\_Id, Insurance name, Ophthalmologist\_Id, Eye examination\_id, Treatment\_Id, and Contact\_Id.

For columns like Patient\_Id, Insurance\_Id, Ophthalmologist\_Id, Eye examination\_id, Treatment\_Id, and Contact\_Id, the INTEGER datatype has been used. By making sure the data is saved in the needed form by using the INTEGER datatype, storage space decreases and query speed is enhanced.

For columns like First name, Last name, Phone number, Username, Password, Specialty, Type of examination, and Treatment type, we've applied the VARCHAR datatype. Variable-length character data, such as names, phone numbers, and text data, can be stored using the VARCHAR datatype. The VARCHAR datatype makes sure that information can be saved correctly while using less storage space.

Data integrity, efficient use of space, and quick query speed are all maintained by careful selection of the datatypes  physical design. In the end, this produces a successful database that meets the requirements of the project and the Hospital

# References

Conceptual, C. (2019) Database design phase 2: Conceptual design, MariaDB KnowledgeBase. Available at: https://mariadb.com/kb/en/database-design-phase-2-conceptual-design/ (Accessed: April 18, 2023).

Inc, C.A. (2023) Introduction to logical design, Introduction to Logical Design. CA INC. Available at: https://techdocs.broadcom.com/us/en/ca-mainframe-software/database-management/ca-idms/19-0/administrating/database-design/introduction-to-logical-design.html#:~:text=idms19-,Logical%20database%20design%20is%20the%20process%20of%20determining%20the%20logical,requirements%20of%20your%20business%20organization. (Accessed: April 18, 2023).

Rick BanisterRick is the Founder and CEO of Sesame Software. He has developed application systems for 40 years in his roles as a product architect, P.D. (2022) Physical Database Design: Sesame Software &amp; Relational Junction, Sesame Software. Available at: https://sesamesoftware.com/white-papers/physical-database-design/ (Accessed: April 18, 2023).