1. Database Requiernment:  
     
   1.1- Scenario:

* My IT company has signed a contract with Royal Prince Alfred Hospital in Australia(Sydney). The company is going to provide a database system in order to help them handle the patient’s data. The system’s goal is to maintain the patient information such as, the medical history, appointments, as well as the healthcare providers. Therefore, our team has been asked to design a database system to meet the hospital and patient needs. The system will be called the “Patient Record and Appointment Management System".

1.2-Data Requirements:

* Royal Prince Alfred Hospital is a facility which provides a wide range of health services to their patients. The database has the following requirements:
* Patients:
* Each patient has their unique identification number, name(first name, last Name), date of birth, age, address(full address), and one phone number.
* Each patient could own multiple medical records.
* Each patient might book multiple appointments.
* Doctors:
* Each doctor has their unique identification number, name(full name), specialty, phone number(at least one), and one email address.
* Each doctor might have multiple appointments.
* Each doctor has one specialty.
* Appointments:
* Each appointment has a unique number, date, and time.
* Each appointment is made by only one patient.
* Every appointment is attended by one doctor.
* Every appointment can be saved in one medical record.
* There might be multiple appointments at the same time but not with the same doctor or patient.
* Medical Records:
* Each medical record has a number, procedure description, medical problem description.
* Each medical record is related to one patient but could be updated by multiple doctors.
* Each medical record could be related to an appointment.
* Department:
* Each department has a number, a name, and which floor it is located in.
* Each department is located on a specific floor (floor).
* Each department can have many appointments associated with it.
* Each appointment can be associated with multiple departments, as a patient may visit multiple departments during a single appointment.
* After each appointment the patient could go to one or multiple departments in the hospital to have different medical examinations, such as medical lab, X-rays, etc.
* Each department is placed on a specific floor in the hospital. And there can’t be two departments in the same floor.
* The administrators will keep track of the number of hours that each patient spends in the department after each appointment.

1.3- The user and system requirement:

* The users of the system are:
* Patients: people seeking health care.
* Doctors: are the health care experts that offer individuals medical attention.
* Nurses: employees who take care of patients and help doctors.
* Administrators: are individuals who are responsible for editing information in the system.
* Functionalities for each user:
  + Patients are able to:
* Schedule doctor’s appointments.
* Review their medical records and make any changes to their personal information.
* Get reminders on their appointments.
  + Doctors are able to:
* View their appointments.
* Access and update their patient’s medical record.
* add medications and treatments to their patients.
  + Nurses can:
* Update the patient’s information based on their condition or any treatments.
* Can view patient information, medical records, and appointments.
* Can update patient medical records.
* Can create and schedule appointments.
  + Administrators can:
* Add, edit, delete a doctor.
* Add, edit, delete a patient.
* Add, edit, delete (patients’ information’s, medical record).

1. Database Design:

2.1- Conceptual Design

* Conceptual design is considered as the beginning step towards database design. The main idea of conceptual design is to create visual design from a certain idea or concept.
* The main goal is to build a model that shows the basic entities, attributes, and the relationships along the system of the database. It is considered a critical step in order to ensure that the database fits the organization’s needs.
* The Components of the conceptual design:
* Entities type: which is the most important part in the system, for example Doctors, patients, appointments, departments, as well as the medical record.
* The cardinality ratios between entities, for example many-to-many.
* Relationship types, for example attend.

Diagram

Description automatically generated

This figure shows the relationship between each entity and the other(Cardinality Ratio). It shows three Cardinality Ratios, one-to-one, one-to-many, and many to many. Also, it shows four relationship types, for example choose, book, saved in, attend.

One-to-one relationship is only between the appointments and medical record entities. Each medical record has one appointment, and each appointment is saved in one medical record.

One-to-many relationship :

* Each doctor can attend many appointments, but each appointment is only attended by one doctor.
* Each patient can book many appointments, but each appointment is only attended by one patient.

Many-to-many relationship :

* Each appointment can choose many departments to visit, and each department visit could be related to one or more appointments.

2.2-Schema and Mapping

* Basic Schema:
* Doctors(Doctor\_id, name, specialty, phone number, email address)
* Patients(Patient \_id, name, birthdate, age, address, phone number)
* Appointments(App\_number, date, and time)
* Medical record(MedRec\_number, procedures\_desc, medical\_problem\_desc)
* Department (dep\_number, name, floor)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Relation name** | **Candidate key(s)** | **Primary Key** | **Prime attributes** | **Non-prime attributes** |
| Doctors | Doctor\_id, phone number, email address | Doctor\_id | Doctor\_id, name, phone number, email address | Specialty.  Name. |
| Patients | Patient \_id, phone number. | Patient \_id | Patient \_id, name, phone number. | birthdate, age, address, name. |
| Appointments | App\_number | App\_number | App\_number | date, time. |
| Medical record | MedRec\_number | MedRec\_number | MedRec\_number | procedures, medical problems. |
| Department | dep\_number, name, floor | dep\_number | dep\_number, name, floor |  |

* First Schema:
* Doctors(Doctor\_id, name, specialty, phone number, email address)
* Patients(Patient \_id, name, birthdate, age, address, phone number)
* Appointments(App\_number, date, and time)
* Medical record(MedRec\_number, procedures\_desc, medical\_problem\_desc)
* Department (dep\_number, name, floor)

**Mapping:**

1. we keep the simple attribute.
2. Remove derived attributes, here we have the age as a derived attribute in the entity type patients. Patients (Patient \_id, name, birthdate, ~~age~~, address, phone number)
3. Split the composite attribute for multiple attributes. Such as the name attribute in patients. Patients (Patient \_id, first\_name, last\_name, birthdate, address, phone number)
4. For multi-valued attributes we make a new relation(entity), it primary key will be the same from the original table and it will be the foreign key also, and finally we add the attribute to it. Such as the phone number for the doctor:

Doctors (Doctor\_id, name, specialty, ~~phone number,~~ email address)

Doctor\_number (Doctor\_id, phone number)

1. No weak entities.
2. In One-to-one relationship we take the primary key from any of the two relations and give it to the other relation as a foreign key:

Appointment saved(stored) in medical record.

Appointments(App\_number, date, and time)

Medical record(MedRec\_number, procedures\_desc, medical\_problem\_desc, App\_number)

1. In one-to-many relationship, we take the primary key from the one cardinality to the many as a foreign key, along with the attributes on the relationship if they exist. We have two 1:N relationship types for attend, book.
2. Doctors(Doctor\_id, name, specialty, email address)

Appointments(App\_number, date, and time)

Appointments(App\_number, Doctor\_id , date, and time)

1. Patients (Patient \_id, first\_name, last\_name, birthdate, address, phone number)

Appointments(App\_number, date, and time)

Appointments(App\_number, Doctor\_id, Patient \_id , date, and time)

1. In many to many relationships, we make a new relation(entity), then we take the primary key from the two tables as a primary key for the new table(relation) with the attributes if existed:

Appointments(App\_number, Doctor\_id, Patient \_id , date, and time)

Department (dep\_number, name, floor)

Dep\_app(App\_number, dep\_number, hours).

* **The schema after mapping:**
  + Doctors(Doctor\_id, name, specialty, email address)
  + Patients (Patient \_id, first\_name, last\_name, birthdate, address, phone number)
  + Doctor\_number (Doctor\_id, phone number)
  + Appointments(App\_number, Doctor\_id, Patient \_id , date, and time)
  + Medical record(MedRec\_number, App\_number ,procedures\_desc, medical\_problem\_desc)
  + Department (dep\_number, name, floor)
  + Dep\_app(App\_number, dep\_number, hours)

Note: Primary keys are underlined, and foreign keys are highlighted in yellow.

2.3-Normalaization:

2.3.1- 1st NF

* The first normal form is a rule in database normalization that ensures every table tuple contains unique values that can’t be divided or broken into smaller parts. The first normal form would be violated if a certain attribute has multi-valued or composite attributes.

|  |  |  |  |
| --- | --- | --- | --- |
| 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. |
| Patients (Patient \_id, name, birthdate, address, phone number) | name | Name in patient is considered as composite attribute therefore it should be broken into first and last name. | Patients (Patient \_id, first\_name, last\_name, birthdate, address, phone number) |
|  |  |  |  |
| Doctors (Doctor\_id, name, specialty, phone number, email address) | phone number | As written in the scenario a doctor could have multiple phone numbers, therefore it is a multi-valued attribute. | Create a new relation for the doctor and phone number. Doctors (Doctor\_id, name, specialty, email address)  Doctor\_number (Doctor\_id, phone number) |

2.3.2- 2nd NF

The second normal form would be achieved if a certain table has achieved the first normal form and every non-prime attribute is not partially dependent on any key. The second normal form would be violated when a non-prime attribute is functionally dependent on a part of the primary key, rather than on the entire primary key.

|  |  |  |  |
| --- | --- | --- | --- |
| 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. |
| App\_dep (App\_number, dep\_number, Doctor\_id, Patient \_id, date, time, name, floor, hours) | Fd1(App\_number, dep\_number)🡪hours  Fd2 (App\_number)🡪 Doctor\_id, Patient id, date, time.  Fd3(dep\_number)🡪 name, floor | Since most of the attributes depend on only a part of the primary key App\_number or dep\_number rather than the entire primary key. | Dep\_app (App\_number, dep\_number, hours).  Appointments(App\_number, Doctor\_id, Patient \_id , date, and time)  Department (dep\_number, name, floor) |
|  |  |  |  |

2.3.3- 3rd NF

Any relation schema is in the third normal form after it has achieved the first normal form and the second normal form, and for non-prime attribute we don’t have any partial dependency, as all the fields should be only determined by the primary key directly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Doctor\_id | App\_number | name | specialty | email address | date | time |

|  |  |  |  |
| --- | --- | --- | --- |
| 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. |
| Doc\_app (Doctor\_id, App\_number, name, specialty, email address, date, and time) | App\_number(A)🡪Doctor\_id(B)  Doctor\_id(B)🡪(name, speciality, email) (C)  App\_number(A)🡪(name, speciality, email) (C) | There is transitive dependency. As (name, speciality, email) is transitively dependent on App\_number through Doctor\_id | Appointments(App\_number, ,Doctor\_id, date, and time)  Doctors (Doctor\_id, name, speciality, email) |
|  |  |  |  |

2.4- Logical Design

* In database design we use the logical design to create a better visual representation of the database.
* The goal behind logical design is to create a schema to show information content of a database, and to show the hospital environment.
* The tables of the logical design will store data that includes the entities of the system, the primary keys, the foreign key’s, attributes and the cardinality ratios.

Diagram

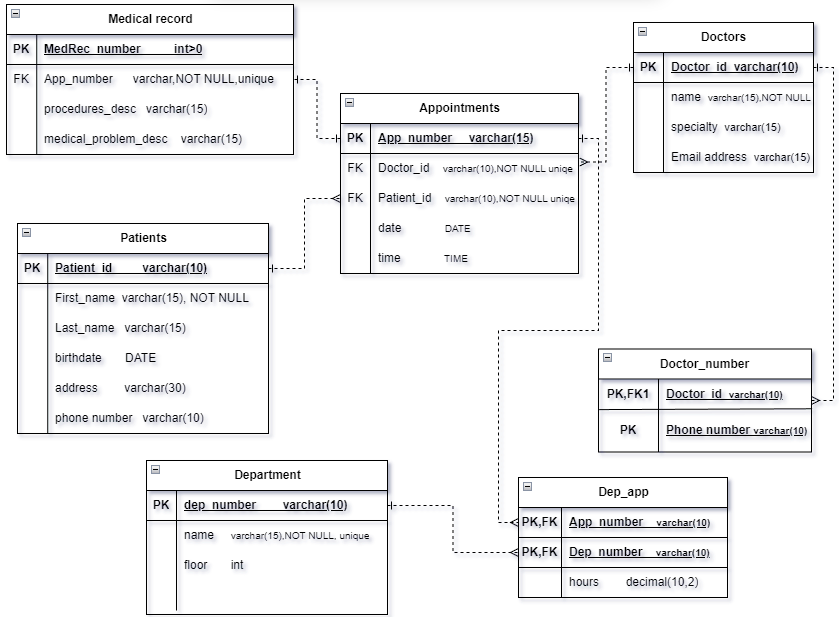
Description automatically generated

This logical design represents a visual representation of the structure of the database system, it includes the tables, attributes, the foreign keys, the primary keys, as well as the cardinality ration between them. Also, it gives a better representation (explanation) of where each foreign key has come from. This figure also represents how the database is organized and how the data is going to be stored in it or retrieved from it.

The entity types such as appointments, Doctors, patients, etc. are shown in the figure with clearly highlighting the primary keys and foreign keys with making sure of the referential integrity which clearly indicates where each foreign key has come from. It also shows the cardinality ratio between each table. It is clear that we have more entities than in the conceptual design such as Dep\_app, and Doctor\_number. And they were also added in the figure and connected to the rest of the entities.

2.5- Physical Design

* The physical design is considered as the last(final) part(stage) in the process of database design. In this part the logical design would be transformed into another form of representation which is physical representation, which later on can be implemented in a database management system for example MySQL, Oracle, Microsoft server. The main idea(purpose) behind physical design is to enhance the performance of databases, storage, and effectiveness.
* The physical design contains entities of the system, the primary keys, the foreign key’s, attributes, cardinality ratios, data types and any constraints on the attributes.



This physical design I made is based on the database system and it is based on the logical design I previously created.

The entity types are shown in the figure also with their tables, attributes, the foreign keys, the primary keys, as well as the cardinality ration between them. But we added the data types and any possible constraints on each table.

2.6- effectiveness of the design

* The effectiveness of the design of a database can be measured by its capability to effectively store and access data, as well as its ability to meet the requirements and demands of the system's users.
* In our case, the design has proven to be effective as it allows doctors to easily view their appointments, access or update their patient’s medical record, as well as adding medications and treatments to be performed for their patient. patients are able to schedule appointments and get reminders for them, as well as reviewing their medical record. Nurses are able to update the patient’s information, creating and scheduling appointments, and viewing patients’ information, appointments, and medical record. Finally, the administrators are able to add, edit, and delete doctors or patients’ information.
* The conceptual design has proven its effectiveness as it successfully and clearly showed the wanted entity types, the cardinality ratio’s as well as the relationship types between the entities. For example, it showed the cardinality ratio for each relationship such as the doctor (attend) appointment in a one-to-many relationship. Or appointments (saved in) medical record in a one-to-one relationship.
* After that came the mapping part which was effective as it worked on each relationship between the entities. In my project I applied the mapping part gradually or by steps. I kept the simple attributes as they are, then removed the derived attribute, after that I split the composite attributes into multiple attributes (name attribute), I made a new relation for the multi-valued attribute with its primary key. I had no week entities, so I started with cardinality ratios, the one-to-one relationship for example I gave a copy of the primary key from the appointments to the medical record as a foreign key. In one-to-many relationship, I gave the primary key from the one entity and gave it to the many entities as a foreign key such as the relationship between doctors and appointments. In many-to-many relationships, I made a new relation and took the primary key from both the entities tables and stored them in a new table along with any attribute on the relationship, such as the relationship between appointments and departments.
* After the mapping I made sure to add the schema that clearly indicates the primary key in each table as well as the foreign key attributes.
* The normalization part was an effective technique in order to make sure that I haven’t forgotten anything in the mapping part, as a matter of fact I haven’t. In the first normal form I repeated the composite and multi-valued rules. In the second normal form I connected the appointment and department tables and found a violation which was later on split into three tables. I did the same for third normal form, I connected Doctor and appointment tables, and I had a violation which was split into two tables.
* The logical design has clearly indicated the entities tables, their primary keys, the foreign keys, and referential integrity. The logical design plays an important part in impacting the performance of the database system. First of all, when designing the logical design correctly it will help later on when implementing the database by making the requesting and retrieving and manipulating easy and fast.
* The physical design has added some extra information on each attribute in each table such as the data types and constraints. Of course, adding the data types and constraints will improve the performance and efficiency of the system. For example, it would help in storage optimization by correctly allocating spaces for the data, as well as ensuring that valid data is being stored in the database. And of course, both of them clearly showed the cardinality ratio between the tables.
* To sum up, the use of foreign keys and normalization techniques has improved data integrity and reduced redundancy, leading to a more reliable and accurate database.

References:

1. MariaDB KnowledgeBase. (n.d.). Database Design Phase 2: Conceptual Design. [online] Available at:  
   <https://mariadb.com/kb/en/database-design-phase-2-conceptual-design/#:~:text=The%20purpose%20of%20the%20conceptual,called%20an%20entity%2Drelationship%20model>.
2. Dr., G. and Riesner, M. (2011). Conceptual Design of Secure Databases. Encyclopedia of Cryptography and Security, Available at:

[https://link.springer.com/referenceworkentry/10.1007/978-1-4419-5906- 5\_673#:~:text=Conceptual%20database%20design%20is%20part,database%20design%20and%20database%20implementation](https://link.springer.com/referenceworkentry/10.1007/978-1-4419-5906-%205_673#:~:text=Conceptual%20database%20design%20is%20part,database%20design%20and%20database%20implementation).

1. Becker, R. (2019). What is Third Normal Form (3NF)? - Definition from Techopedia. [online] Techopedia.com. Available at: <https://www.techopedia.com/definition/22561/third-normal-form-3nf>.
2. Techopedia.com. (2019). What is Second Normal Form (2NF)? - Definition from Techopedia. [online] Available at: <https://www.techopedia.com/definition/21980/second-normal-form-2nf>.
3. CHAPTER 7: LOGICAL DATABASE DESIGN (2019). Fundamentals of Database Management Systems, Second Edition. [online] O’Reilly | Safari. Available at: <https://www.oreilly.com/library/view/fundamentals-of-database/9781118213575/15_chap07.html>.
4. Coursera. (n.d.). Goals and steps of logical database design video lecture - Schema Conversion. [online] Available at: <https://www.mybib.com/tools/harvard-referencing-generator>
5. www.sciencedirect.com. (n.d.). Physical Database Design - an overview | ScienceDirect Topics. [online] Available at: <https://www.sciencedirect.com/topics/computer-science/physical-database-design#:~:text=Physical%20design%20is%20the%20process%20of%20choosing%20specific>
6. www.ibm.com. (n.d.). Db2 11 - Introduction - Physical database design. [online] Available at: <https://www.ibm.com/docs/en/db2-for-zos/11?topic=relationships-physical-database-design>.