[Database design and Development]

[2023]

[Assignment 1 - Report]

ENG: **Lina Hammad**

**[Layan Saeed Aqel] | [21110405]**

**Table of Contents:**

[1 Database Requirement 2](#_Toc132745094)

[1.1 Scenario 2](#_Toc132745095)

[1.2 Data Requirements 2](#_Toc132745096)

[1.3 User and System Requirement 3](#_Toc132745097)

[2 Database Design 4](#_Toc132745098)

[2.1 Conceptual Design 4](#_Toc132745099)

[2.2 Schema and Mapping 4](#_Toc132745100)

[2.3 Normalization 5](#_Toc132745101)

[2.3.1 1st N 5](#_Toc132745102)

[2.3.2 2nd NF 6](#_Toc132745103)

[2.3.3 3rd NF 7](#_Toc132745104)

[2.4 Logical Design 8](#_Toc132745105)

[2.5 Physical Design 8](#_Toc132745106)

[2.6 Effectiveness of the design 9](#_Toc132745107)

[References 12](#_Toc132745108)

# Database Requirement

## Scenario

I am a junior database developer and I work for LS hospital which is a worldwide hospital. I was asked by the manager of the hospital to develop the system of the medicine warehouse in the hospital which has an old system that has been used for 15 years. This system no longer able meet the current expectation of the warehouse. It can’t give clear information about **out of stock** and **orders delay** which are very important for managing job.

LS hospital has many branches, and employees. The new database will have information about the branches, employees, products, warehouse, transaction, costumer information and his order lists.

The new system is a website which allows the warehouse customer (patients) to buy the product of the company. They can see the recent changes related to the products such as the availability, prices, expiration and production date of products.

The new system will solve the out-off-stock problem by putting a reminder when the stock reaches 10% of the overall stock so that doctors can know if any of the products is out-off-stock and so they don’t have it in their patients’ prescriptions. This system will help patient to make sure that the prescribed medicine is available in the workhouse of the hospital.

Doctors can also contact the transportation department if there is any delay in the delivery of the order, which saves time and makes the load on the hospital less. The most important feature of the new system is that it maintains security and privacy so that patients can’t see other patients’ prescriptions.

## Data Requirements

1. Branch (number, name, phone, address (street name, city, building name, building number))
2. Each branch has only one warehouse and each warehouse exist in only one branch.
3. Warehouse (number, name, storage condition, phones)
4. The warehouse has many employees, and each employee works only in one warehouse.
5. Employees (email, phone, addresses (street name, city, building name, building number), salary, name (first name, last name), birthdate, age, level, degree) level represent the degree of employee which A is the highest degree and D is the lowest degree.
6. Customer (id, name (first name, last name), phone, address (street name, city, building name, building number), gender, email, birthdate, age)
7. Product (name, usage, price, form, production date, weight, expiration date, stock)
8. Each customer can do multiple transaction, but transaction only belong to one customer.
9. Transaction (id, amount, channel, number of transaction)
10. The warehouse has different products and each product exist in different warehouses.
11. The one transaction has various product and one product exist in multiple transactions.

## User and System Requirement

* **General manager (CEO): for all warehouses**

1. Can view the employee’s information such as salary, level, and degree.
2. Can view the warehouse information such as storage condition.
3. Generate a report to see detailed data on all products such as the sales and the number of transactions of each warehouse. done
4. Can view the stock of products in any warehouse. Done

* **Warehouse Branch Manager:**

1. Add/update/delete/view the employees’ information in his warehouse such as salary, level, and degree.
2. Can view how many transactions in his warehouse. done
3. Can view how many employees in his warehouse. done
4. Can view how many products in his warehouse. done
5. view the out-of-stock medicines of his warehouse. done

* **Warehouse employees:**

1. Create their own account. done
2. View their information and update some of them such as phone, address, email.
3. Can view data about the available amount of medicine, expiry dates and prices. done
4. Enter data about the product.

* **Customers:**

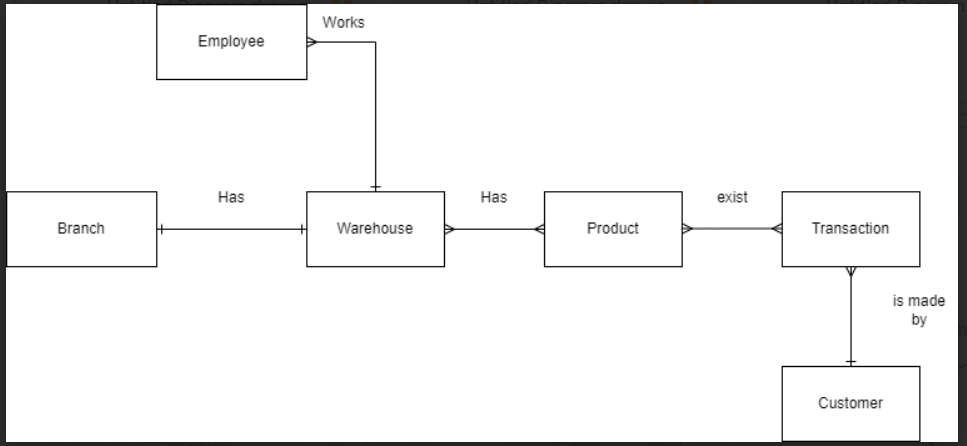
1. Create an account. done
2. view available medicine, prices, and expiry dates. done
3. Do a transaction.
4. update their emails, phones, addresses.

# Database Design

## Conceptual Design

It is the first stage of the process of designing of the project. In this stage, the developer identifies the main entities in the project and the relationships between them. This stage is very important because it makes the main aspects of the project clear such as its vision, concept, and goals.

* My conceptual design depends on my scenario:



In this figure, I identified the main entities of the design of my project which are Branch, warehouse, employee, customer, product, and transaction. Then, I identified the cardinality of the relationships between these entities.

## 2.2 Schema and Mapping

1. **Branch** (Number, name, phone, street name, city, building number, building name)
2. **Warehouse** (Number, name, storage condition, branch-number)
3. Warehouse phone (Warehouse-number, phone)
4. **Employee** (Id, email, phone, salary, first name, last name, birthdate, level, degree, Warehouse-number)
5. Employee address (employee -id, street name, city, building number, building name)
6. **Customer** (Id, first name, last name, phone, gender, email, street name, city, building number, building name, birthdate)
7. **Product** (Id, name, usage, price, form, production date, expiration date, weight, stock)
8. **Transaction** (Id, amount, channel, customer-id)
9. Warehouse product (product-id, warehouse-number)
10. Transaction product (product-id, transaction-id, time, date)

## Normalization

It is an additional step in which we organized data in the database in a correct way. So that we can avoid some problems such as redundancy and complexity of data by using tables to make sure that the attributes and relationships between the entities are correct. Also, we follow three basic rules which are the first, second, third normal form.

### 1st N

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | Attributes | Violation description | Solution – Relations |
| Branch (Number, name, phone, address (city, building number, building name, street name)) | address (city, building number, building name, street name)) | This attribute is composite. So, if I deal with it wrongly will violate first normal form because the cell will have more than one value and it will no longer be atomic which will affect the readability, research of data and increase complexity of data. | **Branch** (Number, name, phone, street name, city, building number, building name) |
| Warehouse (Number, name, storage condition, number of employees, phones) | phones | This attribute is Malti-value. So if I deal with it wrongly will violate first normal form because the row will not be unique or more than one column will have the same name which will affect redundancy and duplication of data. | **Warehouse** (Number, name, storage condition, number of employees)  Warehouse phone (Warehouse-number, phone) |
| Employee (Id, email, phone, salary, name (first name and last name), birthdate, age, level, degree, addresses (city, building number, building name, street name)) | Addresses (city, building number, building name, street name))  Name (first name, last name) | This attribute is complex. So, if I deal with it wrongly will violate first normal form because the row will not be unique and the cell will have more than one value and it will no longer be atomic which will affect the readability, research of data and increase complexity of data. Also, it will affect redundancy and duplication of data.  This attribute is composite. So, if I deal with it wrongly will violate first normal form because the cell will have more than one value and it will no longer be atomic which will affect the readability, research of data and increase complexity of data. | **Employee** (Id, email, phone, salary, first name, last name, birthdate, level, degree)  Employee address (employee -id, street name, city, building number, building name) |

### 2nd NF

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | PDs | Violation description | Solution – Relations |
| Product (product-Id, transaction-id, product-name, product-usage, product-price, product-form, product-production date, product-expiration date, product-weight, product-stock, transaction-amount, transaction-channel, transaction-date, transaction-time) | Product-id -> product-name, product-usage, product-price, product-form, product-production date, product-expiration date, product-weight, product-stock  Transaction-id -> transaction-amount, transaction-channel, transaction-date, transaction-time | In this case we have partial dependency. So, if I deal with it wrongly will violate second normal form because part of the non-key attributes depends on one of the primary key or candidate key.  Which will affect the redundancy and duplication of data. | **Product** (Id, name, usage, price, form, production date, expiration date, weight, stock)  **Transaction** (Id, amount, channel)  Transaction product (product-id, transaction-id, time, date) |
| Warehouse (warehouse-number, product-id, warehouse-name, warehouse-storage condition, product-name, product-usage, product-price, product-form, product-production date, product-expiration date, product-weight, product-stock) | Warehouse-number -> warehouse-name, warehouse-storage condition  Product-id -> product-name, product-usage, product-price, product-form, product-production date, product-expiration date, product-weight, product-stock | In this case we have partial dependency. So, if I deal with it wrongly will violate second normal form because part of the non-key attributes depends on one of the primary key or candidate.  Which will affect the redundancy and duplication of data. | **Warehouse** (Number, name, storage condition)  **Product** (Id, name, usage, price, form, production date, expiration date, weight, stock)  Warehouse product (product-id, warehouse-number) |

### 3rd NF

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | TDs | Violation description | Solution – Relations |
| Employee-transaction (employee-Id, employee-email, employee-phone, employee-salary, employee-first name, employee-last name, employee-birthdate, employee-level, employee-degree, warehouse-name, warehouse-storage condition) | Employee-id ->  warehouse-name -> warehouse-storage condition | In this case we have transitive dependency. So, if I deal with it wrongly will violate third normal form because the attributes in the table will not have direct dependence on its own entity. Also, if we deal with relationships wrongly such as transmitting the foreign key wrongly.  Which will affect the redundancy, indirect dependency and duplication of data. Also, if I want to delete data the transitive dependency makes it more complex. | **Warehouse** (Number, name, storage condition)  **Employee** (Id, email, phone, salary, first name, last name, birthdate, level, degree, Warehouse-number) |
| Customer-transaction (transaction-Id, transaction-amount, transaction-channel, transaction-time, transaction-date, customer- first name, last name, phone, gender, email, street name, city, building number, building name, birthdate) | Transaction-id -> customer-first name-> last name, phone, gender, email, street name, city, building number, building name, birthdate | In this case we have transitive dependency. So, if I deal with it wrongly will violate third normal form because the attributes in the table will not have direct dependence on its own entity. Also, if we deal with relationships wrongly such as transmitting the foreign key wrongly.  Which will affect the redundancy, indirect dependency, and duplication of data. Also, if I want to delete data the transitive dependency makes it more complex. | **Customer** (Id, first name, last name, phone, gender, email, street name, city, building number, building name, birthdate)  **Transaction** (Id, amount, channel, customer-id) |

## Logical Design

It is a more detailed stage in which the developer defines the attributes, primary keys and foreign keys based on the design of the first stage. Also, it defines if the relationships are weak or strong between entities. This stage is very important in defining the logical structure and architecture of the system and the relationships between the entities. After mapping and schema and normalization, the many to many relationships which were in the first stage are changed into one-to-many relationships in this stage. Also, the Malti-value and complex attributes become new entities in this stage.

* My logical design depends on my scenario:

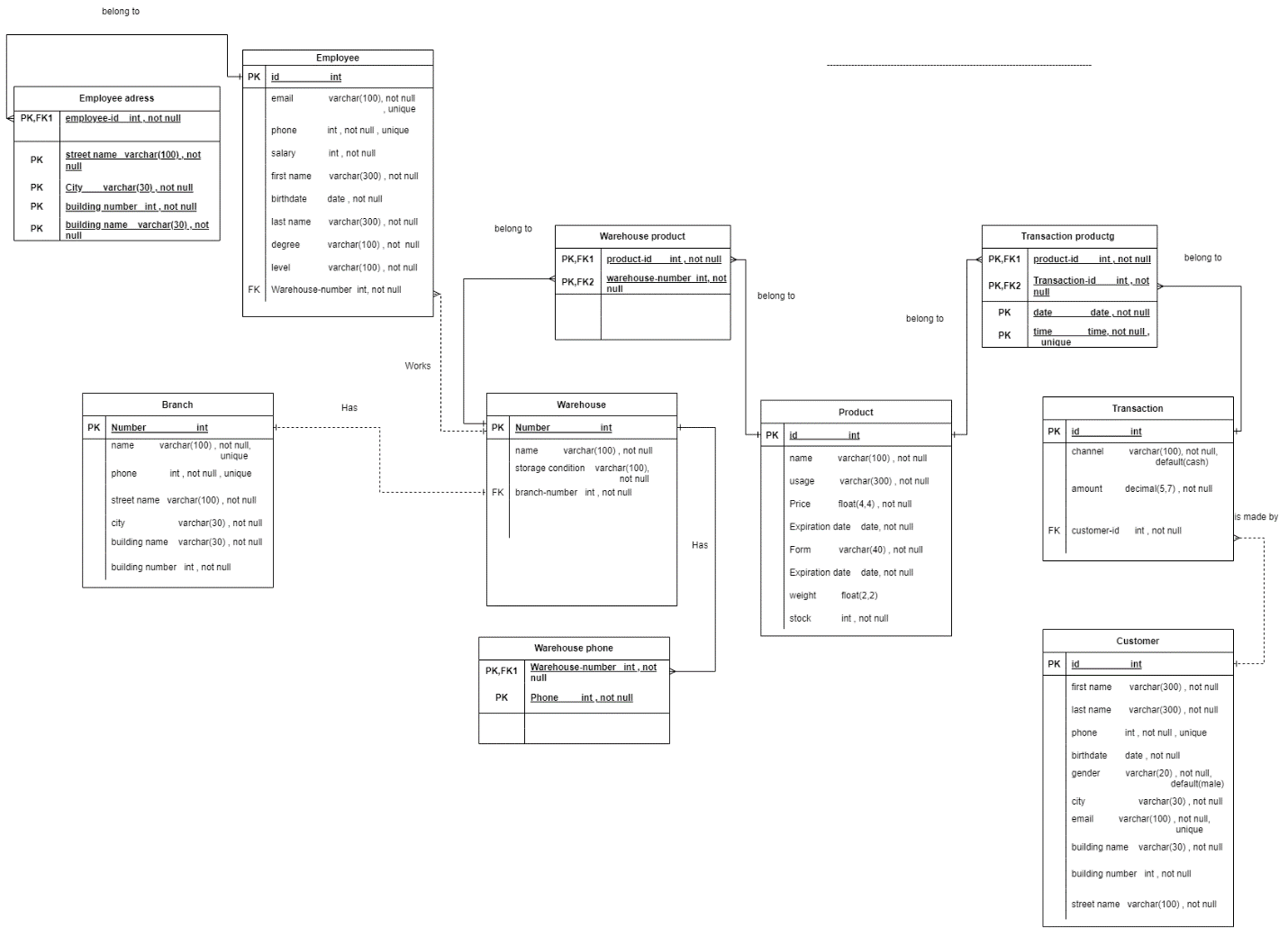
Diagram, schematic

Description automatically generated

## Physical Design

In this stage, the developer adds more details such as data type and constraints to the logical design which shows who data is built in the database.

* My physical design depends on my scenario:



## Effectiveness of the design

I am a database developer and I work for LS Hospital. The manager of the hospital asked to develop the current system of the warehouse of the hospital, which includes designing a database of the warehouse. First, I started by writing a scenario before any other step such as designing the database and optimizing the present system so that I can clearly define the requirements of the new system and define any possible problem so that I can suggest the best solution in agreement with the requirements of the warehouse system. It is true that writing the scenario takes time which may delay business, but the benefits gained on the long term will help me to work in an organised and efficient way and so will save time, effort, and money. I also collected the required data that will be saved in the database in order to help me to know the entities of the database and predict its future design. Another point is building the relationships between the entities. Some of the problems that I found in the last system was the orders’ delay and the out-off-stock. I added some services to deal with these problems. The new system will solve the out-off-stock problem by putting a reminder when the stock reaches 10% of the overall stock so that doctors can know if any of the products is out-off-stock and so they don’t have it in their patients’ prescriptions. This system will help patient to make sure that the prescribed medicine is available in the workhouse of the hospital.

Doctors (employees) can also contact the transportation department if there is any delay in the delivery of the order, which saves time and makes the load on the hospital less. The most important feature of the new system is that it maintains security and privacy so that patients can’t see other patients’ prescriptions. Another helpful of the new system is that the ability to pay either online or cash by using the visa card.

Based on the scenario, I started with the conceptual design which requires defining the entities of the design and the relationships and their cardinalities. The entities are branch, warehouse, employee, customer, product, transaction. I chose those entities in my new scenario because they are the main elements of achieving the objectives of the new design of database. Also, they help me to deal with any optimal problem easily and efficiently because the problem in the last system is related to them. After that, I defined the relationships between these entities.

The relationship between branch and warehouse is that each branch has only one warehouse and each warehouse exists in only one branch, which is one to one cardinality. The relationship between warehouse and employee is that the warehouse has many employees, and each employee works only in one warehouse, which is one to many cardinality. The relationship between customer and transaction is that each customer can do multiple transaction, but each transaction only belongs to one customer, which is one to many cardinality. The relationship between the warehouse and the product is that the warehouse has different products, and each product exists in different warehouses, which is many to many cardinality. The relationship between transaction and product is that each transaction has various products, and each product exists in multiple transactions, which is many to many cardinality.

Then, I started with the mapping and schema which is a step to be done before the logical to make it easy. Mapping consists of the entities and the relationships which are in the conceptual. In this step, I added the attributes to each entity. Attributes are different types, and I will solve it depending on its type. **If it is composite**, I will solve it by deleting the main attribute and keeping all its simple attributes. For example, Name (first name, last name) -> first name, last name.

**If it is Malti value**, I will create a new entity which has the Malti value and primary key of the entity which includes the Malti value which I want to transfer which will be called the foreign key. The primary key of the new entity is both the Malti value and the foreign key. This primary key which consists of these two attributes will be called composite key, but if it consists of one attribute, it’s called a simple key.

**If it is complex**, I will do the two above ways which are composite and Malti value.

**If it is simple**, it will remain the same. **If it is derived**, it will be deleted. Finally, I will define the **primary key** only of the simple attributes under two conditions which are the attribute is unique and it is not repeated (not null). Another important point, I determine the type of attributes depending on the entity itself. For example, I chose the name of warehouse and product as simple attributes because each of them owns one name without more details, but the names of the employees and customers as composite attributes because each name of them is one name but it has more than one part such as first, mid and last name.

I chose the phone of the warehouse as Malti-value attributes because any warehouse may have more than one phone number to receive the requests of patients but the phone of any branch as simple attribute because there is no need for more than one phone number for any branch. There is no need to save some attributes in the database such as the age and number of transactions because we can get them from other attributes such as date of birth and frequency of transaction of customers. Sometimes, the employee may have more than one address, so I used a complex attribute for these addresses, but I used only one address for every customer because it is enough for delivery, so I used a composite attribute for customers’ addresses. At last, I made sure that every entity had a primary key that undertakes the two conditions of unique and not null.

After that, I represented the previous relationships of the conceptual design in mapping and schema. **If the cardinality is many to many**, I will make a new entity which includes two primary keys for the entities which have this relationship. When I transfer primary key from its entity to another entity it will become foreign key. **If the cardinality is one to many**, the entity in the direction of the many will take the primary key for the entity in the direction one. **If the cardinality is one to one**, any entity will take primary key from another entity. Another point in this step is that if the entity used the foreign key as primary key, it will be called weak entity. Otherwise, it will be called strong entity.

The next step is normalization, which is done after mapping and schema. It consists of three steps: first, second and third normalization. The benefit of this stage is to make sure that mistakes in the mapping stage are found and dealt with correctly, so they are solved.

In the first normalization, I make sure that the three types of attributes (composite, complex and Malti value) are done in the first stage and the cell must be atomic and the record (row) is unique and not repeated. Also, each column has a unique name and all the data in each column is of the same type. If I didn’t make the 1st normal form in a correct way, this will negatively affect readability, research of data, and increase complexity of data.

In the second normalization, I checked the partial dependency because If I didn’t make the 2st normal form in a correct way, this will negatively affect redundancy and duplication of data.

The third normalization checks the transitive dependency because If I didn’t make the 3st normal form in a correct way, this will negatively affect redundancy, indirect dependency and duplication of data.

In the logical stage, I will display what did in the mapping stage tables. I will only add the way of connecting entities (tables). At least, if there is one week entity, the relationship between them will be strong. If the entities were strong, the relationship between them will be weak.

The physical phase is the same as the logical phase, but I will add the data type and constraints for each attribute.

After I finished ER model with its three stages I enabled the different users to use the new system for the following uses depending on each user’s position which leads to achieving job efficiently and probably. For example, The CEO can view the employee’s information such as salary, level, and degree, warehouse information such as storage condition and out-of-stock of products while the warehouse branch manager can view the employees’ and warehouse information and out-of-stock of products only of his own branch. This is to maintain privacy and organize work in the warehouse properly. Also, it is a way of defining responsibilities of each user. Another example is allowing employees to make changes related to the amount of medicine, expiry and production dates and prices under the control of their direct manager which makes load less on the manger and keeps contact between each manger and his employees. This will also make employees responsible for making decisions. I give the manger privilege to activate and deactivate customers’ and employees’ accounts to prevent misuse of the system by the customers and employees such as creating more than one account for one employee and customer. Also, I give the employee and customer privilege to change their passwords, phones, emails, and addresses to facilitate job and enable them to do simple and private transactions without direct interference of their manager. At the same time, they can’t change other sensitive data such as the salary, level and degree because these attributes are the responsibility of the manager. I made sure that the sequential hierarchical of the responsibilities of each user is logical, well organized and efficient.

# References

CREATE PROCEDURE update\_customer2(in id\_customer int, in amount int, in channel varchar(200))

update transaction SET amount=amount, channel=channel

WHERE transaction.customer\_id=id\_customer ;

GRANT EXECUTE on PROCEDURE ls\_hospital2.update\_customer2 to 'customer'@'localhost';