Lebanese American University

CSC375

NKIY - Phase IV



A Database Design for GoGreen Plant Nursery

The Team:

Dana Nasser

Wafic Khalife

Miguel Ibrahim

Charbel Abi Younes

December 1, 2021

Table of Contents

[A- Phase IV Updates: 5](#_Toc89175993)

[B- Copyright Notice: 6](#_Toc89175994)

[C- Terms and Conditions: 7](#_Toc89175995)

[D- Background Info: 8](#_Toc89175996)

[E- System Description and Constraints: 9](#_Toc89175997)

[F- Entity Types and Attributes: 11](#_Toc89175998)

[1- Schedule: 11](#_Toc89175999)

[2- Department: 11](#_Toc89176000)

[3- Customer: 12](#_Toc89176001)

[4- Employee: 13](#_Toc89176002)

[5- Branch: 13](#_Toc89176003)

[6- Vehicle: 14](#_Toc89176004)

[7- Greenhouse: 14](#_Toc89176005)

[8- Plant: 15](#_Toc89176006)

[9- Nutrient: 15](#_Toc89176007)

[10- Storage: 16](#_Toc89176008)

[11- Equipment: 16](#_Toc89176009)

[12- Supplier: 17](#_Toc89176010)

[G- Relationships 18](#_Toc89176011)

[H- ER Diagram: 29](#_Toc89176012)

[I - ER to Relational Mapping Algorithms: 30](#_Toc89176013)

[**STEP 1:** Mapping of Regular Entity Types 30](#_Toc89176014)

[**STEP 2:** Mapping of Weak Entity Types 34](#_Toc89176015)

[**STEP 3:** Mapping of Binary 1:1 Relationship Types 35](#_Toc89176016)

[**STEP 4:** Mapping of Binary 1:N Relationship Types 37](#_Toc89176017)

[**STEP 5:** Mapping of M:N Relationship Types 42](#_Toc89176018)

[**STEP 6:** Mapping of Multivalued Attributes 44](#_Toc89176019)

[**STEP 7:** Mapping of N-ary Relationship Types 46](#_Toc89176020)

[**FINAL STEP:** Final Displays 46](#_Toc89176021)

[J- Table Structure for GoGreen Plant Nursery: 50](#_Toc89176022)

[1- DEPARTMENT: 50](#_Toc89176023)

[2- CUSTOMER: 50](#_Toc89176024)

[3- EMPLOYEE: 51](#_Toc89176025)

[4- SCHEDULE: 52](#_Toc89176026)

[5- BRANCH: 52](#_Toc89176027)

[6- VEHICLE: 52](#_Toc89176028)

[7- GREENHOUSE: 53](#_Toc89176029)

[8- PLANT: 53](#_Toc89176030)

[9- SUPPLIER: 54](#_Toc89176031)

[10- STORAGE: 54](#_Toc89176032)

[11- NUTRIENT: 54](#_Toc89176033)

[12- EQUIPEMENT: 55](#_Toc89176034)

[13- DRIVES: 55](#_Toc89176035)

[14- PURCHASES\_FROM: 56](#_Toc89176036)

[15- UTILIZES: 56](#_Toc89176037)

[16- REQUIRES: 56](#_Toc89176038)

[17- CUST\_PURCHASE\_DATES: 57](#_Toc89176039)

[18- EMP\_PHONE\_NUMBER: 57](#_Toc89176040)

[19- SUPP\_SUPPLY\_DATES: 57](#_Toc89176041)

[K- Table Description: 58](#_Toc89176042)

[1- DEPARTMENT: 59](#_Toc89176043)

[2- CUSTOMER: 59](#_Toc89176044)

[3- EMPLOYEE: 60](#_Toc89176045)

[4- SCHEDULE: 60](#_Toc89176046)

[5- BRANCH: 60](#_Toc89176047)

[6- VEHICLE: 61](#_Toc89176048)

[7- GREENHOUSE: 61](#_Toc89176049)

[8- PLANT: 61](#_Toc89176050)

[9- SUPPLIER: 62](#_Toc89176051)

[10- STORAGE: 62](#_Toc89176052)

[11- NUTRIENT: 62](#_Toc89176053)

[12- EQUIPMENT: 63](#_Toc89176054)

[13- DRIVES: 63](#_Toc89176055)

[14- PURCHASES\_FROM: 63](#_Toc89176056)

[15- UTILIZES: 63](#_Toc89176057)

[16- REQUIRES: 64](#_Toc89176058)

[17- CUST\_PURCHASE\_DATES: 64](#_Toc89176059)

[18- EMP\_PHONE\_NUMBER: 64](#_Toc89176060)

[19- SUPP\_SUPPLY\_DATES: 64](#_Toc89176061)

[L- Inserting data: 65](#_Toc89176062)

[1- DEPARTMENT: 65](#_Toc89176063)

[2- CUSTOMER: 66](#_Toc89176064)

[3- EMPLOYEE: 67](#_Toc89176065)

[4- SCHEDULE: 69](#_Toc89176066)

[5- BRANCH: 70](#_Toc89176067)

[6- VEHICLE: 71](#_Toc89176068)

[7- GREENHOUSE: 71](#_Toc89176069)

[8- PLANT: 72](#_Toc89176070)

[9- Storage: 74](#_Toc89176071)

[10- Nutrient: 75](#_Toc89176072)

[11- Supplier: 76](#_Toc89176073)

[12- Equipment: 76](#_Toc89176074)

[13- Drives: 78](#_Toc89176075)

[14- PURCHASES\_FROM 78](#_Toc89176076)

[15- UTILIZES 79](#_Toc89176077)

[16- REQUIRES: 80](#_Toc89176078)

[17- CUST\_PURCHASE\_DATES: 80](#_Toc89176079)

[18- EMP\_PHONE\_NUMBER: 81](#_Toc89176080)

[19- SUPP\_SUPPLY\_DATES: 82](#_Toc89176081)

[M- Final Tables State: 84](#_Toc89176082)

[1- DEPARTMENT: 84](#_Toc89176083)

[2- CUSTOMER: 84](#_Toc89176084)

[3- EMPLOYEE: 85](#_Toc89176085)

[4- SCHEDULE: 85](#_Toc89176086)

[5- BRANCH: 86](#_Toc89176087)

[6- VEHICLE: 86](#_Toc89176088)

[7- GREENHOUSE: 87](#_Toc89176089)

[8- PLANT: 87](#_Toc89176090)

[9- STORAGE: 88](#_Toc89176091)

[10- EQUIPMENT: 88](#_Toc89176092)

[11- PURCHASED\_FROM: 89](#_Toc89176093)

[12- DRIVES: 89](#_Toc89176094)

[13- SUPPLIER: 90](#_Toc89176095)

[14- REQUIRES: 90](#_Toc89176096)

[15- NUTRIENT: 91](#_Toc89176097)

[16.UTILIZES: 91](#_Toc89176098)

[17. CUST\_PURCHASE\_DATES: 92](#_Toc89176099)

[18.EMP\_PHONE\_NUMBER: 92](#_Toc89176100)

[19. SUPP\_SUPPLY\_DATES: 93](#_Toc89176101)

[N- Queries: 94](#_Toc89176102)

[Query 1: 94](#_Toc89176103)

[Query 2: 95](#_Toc89176104)

[Query 3: 95](#_Toc89176105)

[Query 4: 96](#_Toc89176106)

[Query 5: 97](#_Toc89176107)

[Query 6: 98](#_Toc89176108)

[Query 7: 99](#_Toc89176109)

[Query 8: 100](#_Toc89176110)

[Query 9: 101](#_Toc89176111)

[Query 10: 102](#_Toc89176112)

[Query 11: 103](#_Toc89176113)

[O- Normalization Up to The BCNF Normal Form: 104](#_Toc89176114)

[1- DEPARTMENT: 106](#_Toc89176115)

[2- CUSTOMER: 107](#_Toc89176116)

[3- EMPLOYEE: 108](#_Toc89176117)

[4- SCHEDULE: 109](#_Toc89176118)

[5- BRANCH: 110](#_Toc89176119)

[6- VEHICLE: 111](#_Toc89176120)

[7- GREENHOUSE: 112](#_Toc89176121)

[8- PLANT: 113](#_Toc89176122)

[9- NUTRIENT: 115](#_Toc89176123)

[10- STORAGE: 116](#_Toc89176124)

[11- EQUIPMENT: 117](#_Toc89176125)

[12- SUPPLIER: 118](#_Toc89176126)

[13- PURCHASES\_FROM: 119](#_Toc89176127)

[14- REQUIRES: 120](#_Toc89176128)

[15- DRIVES: 121](#_Toc89176129)

[16- UTILIZES: 121](#_Toc89176130)

[17- CUST\_PURCHASE\_DATES: 121](#_Toc89176131)

[18- EMP\_PHONE\_NUMBER: 121](#_Toc89176132)

[19- SUPP\_SUPPLY\_DATES: 121](#_Toc89176133)

[P- Conclusion: 122](#_Toc89176134)

[Q- Instructor’s Comments and Evaluation: 123](#_Toc89176135)

# Phase IV Updates:

* Normalize the database up to the BCNF Normal Form.
* Explain the normalization process.

# Copyright Notice:

Copyright ©️ 2021-2022. NKIY LTD. This design and report are a copyright of NKIY LTD. This work is protected by Copyright, and consent should be obtained from the publisher before any excluded reproduction, storage in a recovery scheme, or transmission in any way or by any means, electronic, mechanical, photocopying, or recording. To acquire permission(s) to use material from this work, you may ask or submit a written request to one of the creators of this masterpiece:

[wafic.khalife@lau.edu](mailto:wafic.khalife@lau.edu),

[charbel.abiyounes02@lau.edu](mailto:charbel.abiyounes02@lau.edu),

[dana.nasser01@lau.edu](mailto:dana.nasser01@lau.edu) ,

[miguel.ibrahim@lau.edu](mailto:miguel.ibrahim@lau.edu).

# Terms and Conditions:

-You vow to keep this information safe and secure and not allow any leakage.

-You vow not to copy this information and use it with your database design or any database design.

-You vow not to destroy, incinerate, tear, and even feed it to your dog.

-You may use this report for the use of public good, but never for self-benefit, and even in the public good (government), you must keep aware of the people that you are dealing with.

-The information disclosed to you is for Dr. Ramzi Haraty alone (maybe viewed with family members and pets alone at home but not in public)

Name: Signature

\_\_\_\_ \_\_\_\_\_

# Background Info:

NKIY or, in short, Nasser Khalifeh Ibrahim Younes database management systems to data manage several plant nurseries locally. The idea or goal of our team is to help plant nurseries in repopulating our planet Earth with rare and threatened to be extinct species of plants, flowers, and much more.

In our report, we summarize what we have done during our work process. We were assigned to create a database design for GoGreen nursery. Overall, our work experience has been constructive and useful for our careers. This report will explore the different aspects of what we have done when designing the database for this local nursery, as they were preparing to expand.

In short, our goal is to help this local nursery achieve its goal by designing database systems with the idea of keeping things simple for GoGreen’s employees, departments, and customers. We would cordially thank GoGreen nursery for giving us the chance to be part of this amazing project.

# System Description and Constraints:

The importance of plants to humans and just about all other life on Earth is staggering. Plants bring life to not only our company but the earth as well, so it is vital to keep repopulating planet Earth with plants as we head to a more technological and possibly environmental threatening future. Protecting and taking care of plants in recent times should be a priority for several organizations and not only our company. GoGreen plant nursery is continuously expanding and repopulating our planet with trees and plants that were once lost due to natural and unnatural processes while providing customers with a wide spectrum of plants from fruitless to trees.

GoGreen’s staff consists of hundreds of botanists and agricultures, who make sure that plants are taken care of, protected, and raised to a particular age. In addition, there exist office employees that ensure everything is organized and well equipped for the plants, farmers, and customers. Also, all employees in this nursery are provided with schedules that will help prevent time conflicts and reduce time management errors.

- Manager of the Agricultural Department

- CEO

- Intern (pat and pending)

- Greenhouse supervisor

- Head of Customer Service

- Botanist

- Farmer

The primary function of this nursery is to assist plants and grow them until they reach a particular age, and then they are sold at GoGreen’s branches in pots to be later grown in fields or remained in pots. This nursery must be made up of multiple departments to ensure that the previous process is accomplished smoothly. Each Department has its responsibility to manage the various aspects of the plant nursery such as employees, greenhouse, vehicle, branch, and many more and may as

- Agricultural department

- Customer Service/Support

- Branch Manager

- HR

- Transportation Management

GoGreen nursery is nothing without its greenhouses. Greenhouses are distributed in several regions and come in different types such as freestanding, attached as well as different sizes of land. In these greenhouses, we have storage that store nutrient and equipment required for the growth of the plants.

The normal course of events in the nursery revolves around plants. Plants can either be provided by suppliers or directly grown in one of the greenhouses. Once a plant enters this nursery, it is assigned a code to uniquely identify her. After that, the plant will be in a particular greenhouse where it will be taken care of using a certain dosage of fertilizers, pesticides, and nutrients. Once it reaches a specified age, this plant will be moved using one of our vehicles or trucks to the nearest branch, where it will be sold to either local farmers or citizens.

A plant might need a different amount of nutrients, and some don’t need any nutrients at all; such plants require as well as different dosages of the nutrient as follows:

-Rosemary needs a 50ng (nano-grams) of Nitrogen-Magnesium fertilizers.

GoGreen also takes into consideration the client's point of view. It has built branches in several regions to reduce customers’ traveling expenses in these difficult times. Customers can also contact the Customer Support department in case of any problem during their purchases and can also contact the agricultural department for tips or guides on how to take care of their plans.

# Entity Types and Attributes:

## Schedule:

Diagram

Description automatically generated with medium confidence

The SCHEDULE is an entity type mainly for the employees, and it is the weak entity type of this database. It has the start and end time of their shift. All employees have their independent SCHEDULE.

## Department:

Diagram

Description automatically generated

Departments can work more like well-oiled machines, focusing time and energy on productive tasks. Thus, an entity type DEPARTMENT was needed to increase productivity and avoid chaos in the company. This entity type is uniquely determined by a number of 2 digits and by its name. For example, we can have Sales, Agriculture, HR departments. The DEPARTMENT has a location which is composed of city, street number, building, floor number. It also has an email, phone number to promote communications between departments and other entities. For instance, employees can easily contact departments in case they face a certain problem.

## Customer:

Diagram

Description automatically generated

Customers are the people that purchase plants from the branches of the nursery. After purchasing an item, the CUSTOMER gets a unique ID of 8 digits which is considered the key attribute of this entity type. Naturally, every CUSTOMER has a name. This entity type takes the customer's phone number and address (composite attribute) as attributes to contact or pass by the customer if any problem occurs. Also, the purchase dates will be recorded to help the branches track the transactions that occurred between the branch and customers.

## Employee:

Diagram

Description automatically generated

Employees aren't just workers. They're the backbone of this business. They are responsible for taking care of our plants, greenhouses, branches, and even departments. Hence, creating an EMPLOYEE entity type is a necessity for this database. This entity type records the employee's personal information. It keeps track of the name, which is composed of first and last name, gender, date of birth, a derived attribute age, phone number which may have several values (home, mobile), and an address composed of the city, street name, building and floor. Each EMPLOYEE has a unique ID formed of 10 digits to uniquely identify each employee, and a specialization in the company, for example, gardener, plant biologist, or simply a driver. Finally, employees receive monthly salaries to reward them for their hard work.

## Branch:

Diagram

Description automatically generated

The BRANCH is a shop where all the plants are sold at. Every BRANCH has its unique number formed of 2 digits. It is also uniquely determined by its location (composed of city and street numbers) since we only have one branch in a particular area. All BRANCHs also have a phone number and email to be easily contacted with.

## Vehicle:

Diagram

Description automatically generated

The VEHICLE is used to transport plants from different locations or to transport equipment. The key is the registration number. The registration number will allow the departments to track the VEHICLE during transportation. Another key attribute is the serial number. We also store in our database the type of VEHICLE. The type will allow us to track the vehicles even faster, so we have fewer vehicles to sort and filter through. This will also allow us to know what vehicles to use according to the items transported. For example, if the plants are large, we would require large trucks.

## Greenhouse:

Diagram

Description automatically generated

A GREENHOUSE is a place in which plants are taken care of. Each GREENHOUSE has its very own number made of 2 digits; therefore, it represents the key attribute of this entity type. Greenhouses come with different types such as freestanding, attached as well as different sizes of land. This entity type keeps track of the number of plants available in the GREENHOUSE, which is a derived attribute, along with its location, which consists of the city and street name.

## Plant:

Diagram

Description automatically generated

Plants are the main focus of this nursery. They have different species and names. Once stored in our nursery, a unique code, composed of 4 digits, is assigned to each PLANT to differentiate them. This entity type keeps track of the age of the PLANT and its water usage to inform botanists and even customers about the amount of water needed. We also keep the price of the PLANT if we want to sell them at one of our branches.

## Nutrient:

Diagram

Description automatically generated

Nutrients are essential in this nursery since plants can't grow effectively without them. Each NUTRIENT has a name, and once stored in our nursery, a unique code, formed of 4 digits, is assigned to keep track of them. This entity type also has an expiry date and a derived attribute time to expire that informs the agriculture department and botanists about the products that are close to being expired. We also store the cost of nutrients to keep track of financial records and to be able to approximate the market price of those products.

## Storage:

A screenshot of a phone

Description automatically generated with low confidence

The STORAGE entity type represents the place where equipment and nutrients are stored. It is characterized by a storage number, made of 2 digits, as well as a maximum capacity attribute.

## Equipment:

Diagram

Description automatically generated

Equipment is necessary to the plant nursery. They are stored in the storage rooms and used in the greenhouse when needed. The primary key of the EQUIPMENT in our database is the serial number. Each kind of EQUIPMENT has a serial number that we can store and look at when needed. Using the serial number, the employees will know more detail about the tools needed. Other keys include the type of equipment, status, cost, and quantity. The type of equipment will allow the employees to categorize the EQUIPMENT properly and organize them according to the similarity in the storage rooms. The status in our database will let the employees know if the EQUIPMENT is being used, available for use, or even under maintenance.

## Supplier:

Diagram

Description automatically generated

Suppliers are the ones who provide us with equipment, resources, and even plants. They supply the plant nursery with the essential material to keep working. The primary key is the supplier’s ID. The ID of the supplier is a unique number of 9 digits given to each SUPPLIER and lets the employees know which SUPPLIER supplied what. Our database also keeps track of the suppliers’ names. The name will allow the employees to track the whole process and the transactions. This entity type also includes attributes such as phone numbers and supply dates. The phone number is another key attribute as it allows the employees to contact the SUPPLIER if anything goes wrong, and supply dates are stored to know when exactly these transactions occurred.

# Relationships

A picture containing text, clipart

Description automatically generated

Vehicles are used to transport plants in this nursery. Therefore, a relationship called "Drives" between the EMPLOYEE and VEHICLE entity type has been made. All vehicles are driven by employees, but not all employees drive vehicles. Thus, the participation is total from one side, which is the vehicle's side, and is partial from the other side. Also, many employees drive many vehicles, which explains the many-to-many relationship.

A picture containing text, clipart

Description automatically generated

Every employee has one schedule assigned to him. Thus, an identifying relationship “Is\_appointed” has to be made between the weak entity type SCHEDULE and EMPLOYEE entity type. The participation is total on both sides since all employees have schedules and all schedules are assigned to employees. Moreover, each employee has one schedule assigned to him, and a schedule is given to only one employee. Therefore, this relationship is one-to-one.

A picture containing text, clipart

Description automatically generated

Due to their importance, departments need to be supervised and managed by the responsible employee. Thus, a "Manages" relationship has to occur between the DEPARTMENT and EMPLOYEE entity types. One Employee can manage one department and a department can have one manager. Therefore, it is one to one relationship. Moreover, every department needs to be managed but not all employees supervise departments; thus, there is total participation only from the department side.

Diagram

Description automatically generated

A supervisor is needed to supervise the work of other employees. This explains why we have a self-referencing relationship “Supervises” on EMPLOYEE. One employee may be the supervisor of many other employees; hence, this is a one-to-many relationship. In addition, not every employee is supervised, and not every employee is a supervisor; hence, we have partial participation of the two entity types in this relationship.

A picture containing text, clipart

Description automatically generated

The division of employees into departments allows them to specialize. Hence, a relationship “Works\_for” is needed to connect EMPLOYEE to DEPARTMENT. Many employees can work in a single department; thus, we have a many-to-one relationship between the two entity types. Since every employee belongs in a department and every department contains employees, we have total participation of both entity types in this relationship.

Diagram

Description automatically generated

Customers can purchase well-treated plants from the branches. Thus, a "Purchases\_from" relationship has to occur between the CUSTOMER and BRANCH Entities. All customers buy from branches and all branches sell to customers. Therefore, there is total participation on both sides of this entity type. Moreover, many customers buy from many branches. Thus, a many-to-many relationship is presented. This relationship has an attribute called quantity purchased which purpose is to specify the number of items that the customer has bought from the branch.

Diagram

Description automatically generated

A manager is an essential component of a branch. This explains why we need a relationship “Is\_managed\_by” between BRANCH and EMPLOYEE. Since all branches have a manager but not all employees are branch managers, we only total participation of BRANCH in this relationship. Also, a branch has one manager, and an employee can manage at most one branch hence, we have a one-to-one relationship.

A picture containing text, clipart

Description automatically generated

Many plants can be stored in branches, hence, creating a many to one relationship “Is\_Stored” between PLANT and BRANCH was a necessity. We have total participation of BRANCH since all branches store plants. However, not all plants are stored in branches since they might be at a greenhouse, hence, we partial participation of PLANT in this relationship.

A picture containing text, clipart

Description automatically generated

Each vehicle is used to transport many plants in our nursery. Thus, the one-to-many relationship "Transports" must be created between VEHICLE and PLANT entity types. We have total participation of vehicles in this relationship because all vehicles in our nursery are used to transport plants only. However, not all plants are being transported since some remain stored. Hence, plants participate partially in this relationship.

A picture containing text, clipart

Description automatically generated

Many employees can work in a greenhouse. Hence, the many to one relationship "Works\_in" between EMPLOYEE and GREENHOUSE was needed. Not all employees work in a greenhouse, but all greenhouses have employees working in them. This justifies why we only have total participation of GREENHOUSE in this relationship.

A picture containing text, clipart

Description automatically generated

Equipment is being utilized by the employees working in greenhouses. Thus, a “Utilizes” relationship between the entity types EMPLOYEE and EQUIPMENT is needed. Not all equipment is used by employees since some are left to work alone like automatic water drills, also not all employees use equipment. Moreover, several types of equipment can be utilized by many employees: therefore, a many-to-many relationship.

Diagram

Description automatically generated

A greenhouse needs to be well equipped with several types of equipment like water drills and other machines to store and help plants grow. Thus, a one-to-many relationship “Needs” was created to connect GREENHOUSE and EQUIPMENT. All greenhouses have different types of equipment to take care of plants, and all equipment is used in a greenhouse, thus we have total participation on both sides.

Diagram

Description automatically generated

Many plants might be located in a greenhouse where they are taken care of. This justifies the need for a many-to-one relationship “Is\_located” between PLANT and GREENHOUSE. Like we previously said plants can be stored in branches hence, not all plants are located in a greenhouse. However, all greenhouses contain plants, hence, we have a partial participation from PLANT, but total participation of GREENHOUSE is this relationship.

Diagram

Description automatically generated

A plant may require nutrients, and nutrients can be given to many different plants; thus, the many-to-many relationship "Requires" between PLANT and NUTRIENT was needed. Plants don't totally participate in this relationship since not all plants require nutrients. However, all nutrients in our nursery are used on plants. Hence, we have total participation of nutrients in this relationship. This relationship has an attribute called dosage which purpose is to specify the quantity of a nutrient required by the plant.

A picture containing text, clipart

Description automatically generated

A greenhouse needs to have one or many storages. Therefore, a "has\_a" relationship occurs between the STORAGE and GREENHOUSE entity type. Each greenhouse individually has many storages; therefore, a one-to-many relationship occurs between the two. Moreover, there is total participation on both sides because all greenhouses have storage, and all stores are located in greenhouses.

Diagram

Description automatically generated

Storage is the place where nutrients might be stored. Thus a "Stores" relationship was needed between STORAGE and NUTRIENT. Storages store many nutrients, but not all of them are stored and not all storages contain nutrients; therefore, total participation is not occurring on both sides. One storage can store many nutrients; thus, a one-to-many relationship occurs between the two. This relationship has an attribute called quantity stored which purpose is to specify the quantity of a nutrient stored in each storage.

A picture containing text, clock

Description automatically generated

Storages might also contain equipment. Thus, a “contains” relationship is needed between the STORAGE and EQUIPMENT entity types. storages contain several types of equipment, but not all of them are stored and not all storages contain equipment; therefore, total participation is not occurring on both sides. One storage can store several pieces of equipment; thus, a one-to-many relationship occurs between the two.

A picture containing text, clipart

Description automatically generated

Suppliers supply the plant nursery with nutrients. Thus, a relationship named "Supplies\_1" must be made between the NUTRIENT and SUPPLIER entity types. Many nutrients can be supplied by suppliers. Therefore, many-to-many relationship occur between the two. Furthermore, all nutrients are supplied by a supplier, and not all suppliers supply nutrients; thus, a total participation occurs from one side. This relationship has an attribute called quantity supplied which is the number of nutrients supplied by the supplier.

A picture containing text, clipart

Description automatically generated

Suppliers also supply the plant nursery with equipment. Thus, a relationship named "Supplies\_2" must be made between the EQUIPMENT and SUPPLIER entity types. Several types of equipment can be supplied by suppliers. Therefore, many-to-many relationship occur between the two. Furthermore, a piece of equipment is supplied by a supplier, and not all suppliers supply equipment; thus, a total participation occurs from one side.

A picture containing text, clipart, businesscard

Description automatically generated

The supplier provides seeds of plants. Therefore, a relationship called “provides” is needed between the PLANT and SUPPLIER entity types. this relationship is considered one too many because one supplier provides many plants. And there is no total participation in this relationship because not all suppliers provide plants and not all the plants are provided by a supplier.

# Diagram Description automatically generated ER Diagram:

# I - ER to Relational Mapping Algorithms:

After designing the abstract and simple mapping of the data structure to an ER diagram, the next step is to transform them into relational database design which requires a seven-step procedure. The following is a detailed description of applying the different steps to our database design.

## **STEP 1:** Mapping of Regular Entity Types

In the first step, the regular entity types must be mapped into relations. Each regular entity is going to have its relation that includes all of its simple attributes and a single primary key that is underlined. The regular (strong) entities in this database design are DEPARTMENT, CUSTOMER, EMPLOYEE, BRANCH, VEHICLE, GREENHOUSE, PLANT, NUTRIENT, STORAGE, EQUIPMENT, and SUPPLIER.

1. **DEPARTMENT:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Department-  number | Name | Email | Phone\_  number | City | Street\_  Nb | Building |
| Floor |

The DEPARTMENT entity type contains simple, and composite attributes in addition to a primary key which is underlined, for this case we will be using *‘Department\_number’* as our main primary key. The composite attribute location that has *‘Street\_Nb’*, *‘Building’*, *‘Floor’*, and *‘City’* are included in the relation.

1. **CUSTOMER:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Phone\_number | Name | City | Street\_Nb |

This entity type CUSTOMER contains the primary key *‘ID’*, simple attributes *‘Name’* and *‘Phone\_Number’* (which was originally a candidate key), and a composite attribute address that has *‘City’* and *‘Street\_Nb’*. This entity type also has a multi-valued attribute that won’t be added here due to it being part of another step.

Something we didn’t mention here but is also vital is that the customer is always right.

1. **EMPLOYEE:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | F\_Name | L\_Name | Gender | Salary | DOB | Age |
| Specialization | City | Street\_Nb | Building | Floor |

The biggest and most confusing entity type is the EMPLOYEE. It contains simple, multi-valued, derived composite and primary attributes of which: *‘Phone\_Number’* is the multi-valued (not included here since it's not part of step 1), *‘Age’* that is derived, Name that is composite of *‘L\_Name’*, *‘F\_Name’*, Address is also composite of *‘City’*, *‘Street\_Nb’*, *‘Building’*, *‘Floor’*, and *‘ID’* being the main primary key.

Note: Age = Date now – Year of Birth.

1. **BRANCH:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Branch\_number | City | Street\_number | Email | Phone\_number |

In this entity type, there exists a primary key *‘Branch\_Number’*, simple keys: *‘Email’*, and *‘Phone\_number’*. The composite attribute location has been reduced to both *‘City’* and *‘Street\_number’* for this step.

1. **VEHICLE:**

|  |  |  |
| --- | --- | --- |
| S/N | Registration | Type |

Like we previously said this entity type is used to move plants from the greenhouse to the branch. In this entity type, we have *‘S/N’*, which is a primary key, *‘Registration’* and *‘Type’* as simple key attributes.

1. **GREENHOUSE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Greenhouse-number | City | Street\_name | Type | Size | Number\_of\_plants |

In this entity type we have a primary key *‘Greenhouse\_number’*, derived key attribute *‘Number\_of\_plants’*\*, a composite attribute location that has been sub-divided into both *‘City’* and *‘Street\_name’*, finally, we have simple attributes *‘Size’* and *‘Type’*.

Note: The number of plants depends on the size of the plant and how well we can efficiently organize the space to fit the plants.

1. **PLANT:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Code | Species | Name | Price | Age | Water\_usage |

No plant nursery is a plant nursery without its plants. In this entity type, plants have different attributes. The primary attribute *‘Code’* and the simple attributes such as its *‘Name’*, *‘Age’*, ‘*Species*’, ‘*Water\_usage’* and *‘Price’*.

1. **NUTRIENT:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Code | Name | Cost | Expiry\_date | Time\_to\_Expire |

This entity type consists of its primary key *‘Code’*, the simple keys *‘Name’*, *‘Cost’* and *‘Expiry\_date’*, and it consists of a derived attribute *‘Time\_to\_expire’*\*.

\*: To calculate the time to expire, we run the expiry date with the current date of time, if Expiry date ex: 10/23/21 and today is 10/23/21 then it is expired --> 0.

1. **STORAGE:**

|  |  |
| --- | --- |
| Storage\_number | Maximum\_capacity |

It is the smallest entity type in this company. The storage consists of two attributes, one primary attribute *‘Storage\_number’*, and one simple attribute *‘Maximum\_capacity’*.

1. **EQUIPMENT:**

|  |  |  |  |
| --- | --- | --- | --- |
| S/N | Type | Cost | Status |

The equipment helps increase the harvest while keeping things efficient. It is given a primary key attribute called ‘*S/N* ‘(serial number in short), and simple attributes ‘*Type*’, ‘*Quantity*’, ‘*Status*’, and ‘*Cost*’.

1. **SUPPLIER:**

|  |  |  |
| --- | --- | --- |
| ID | Name | Phone\_number |

Finally, we have reached the end of Step 1, the final entity type to be translated is SUPPLIER. This entity type has a multi-valued attribute ‘*Supply\_dates’* that will be mentioned in step 6 of the translation, a simple attribute *‘Phone\_Number’*, and *‘Name’*, and a key attribute *‘ID’*.

## **STEP 2:** Mapping of Weak Entity Types

In this step, the weak entity types are mapped into relations. As in Step 1, only the simple attributes are included in the relations. Furthermore, weak entity type relation has a foreign key attribute which is the primary key of the owner entity type. The combination of the foreign key added, and the partial key of the weak entity type represents the primary key of the relation. The weak entity type in our database design is SCHEDULE.

1. **SCHEDULE:**

|  |  |  |
| --- | --- | --- |
| Emp-ID | Start-time | End-time |

The weak entity type SCHEDULE has three simple attributes which are *‘Start\_time’*, *‘End\_time’*, and *‘Hours\_worked’*. It does not have a multi-valued nor a derived attribute. Moreover, the primary key of the owner entity type *‘Emp-ID’* is included. *‘Emp-ID’* and the partial key *‘Date’* are combined to represent the primary key of this relation.

## **STEP 3:** Mapping of Binary 1:1 Relationship Types

In this step, we are going to map the binary one-to-one relationships. To accomplish our goal, we can follow one of three approaches. The first approach, called the foreign key approach is where we choose the entity type on the total participation side of the relationship, then we add as a foreign key the primary key of the other entity type participating in this relationship. The second approach, called the merged relation approach is where we merge the two entities participating in the relationship into a single relation. This is only used when both participations are total. The third approach, called cross-reference or relationship relation approach is where we create a third relation which will include the primary keys of both entities participating in the relationship. The binary one-to-one relationships that need to be mapped are: Is\_appointed, Manages, and Is\_managed\_by. In Is\_appointed we have total participation on both sides; hence, we will use the merged approach to map it. For the others, we are going to follow the foreign key approach because it is the most useful in our case.

1. **DEPARTMENT (Manages):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Department-  number | Name | Email | Phone\_  number | City | Street\_Nb | Building |
| Floor | EMP-ID |

Each employee manages different departments. The "MANAGES" is a relationship between the EMPLOYEE entity type and the DEPARTMENT entity type. Here, we don't have full participation between the entity types. Thus, the approach followed here is the foreign key approach. The full participation in the department entity type, which explains why we used the relation DEPARTMENT. In this case, we added, as a foreign key, the primary key ID of the employee entity type and named it *‘EMP-ID’*.

1. **BRANCH (Is\_managed\_by):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Branch\_number | City | Street\_number | Email | Phone\_number | EMP-ID |

Each branch is managed by different employees. The “IS\_MANAGED\_BY” is a relationship between the EMPLOYEE entity type and the BRANCH entity type. Here, we don’t have full participation between the entity types. Thus, the approach followed here is the foreign key approach. The full participation is the BRANCH entity type, which explains why we used the relation BRANCH. In this case, we added, as a foreign key, the primary key ID of the employee entity type and named it *‘EMP-ID’*.

## **STEP 4:** Mapping of Binary 1:N Relationship Types

In this step, we are going to map the binary one-to-many relationships. We add a foreign key in the entity type at the many sides of the relationship. This foreign key is the primary key of the other entity type participating in this relationship. We must also include any other simple attribute of the one-to-many relationship. The one-to-many relationships that need to be mapped are: Supervises, Works\_for, Is\_stored, Transports, Works\_in, Needs, Is\_located, Has\_a, Stores, Contains, Supplies\_1, Supplies\_2, and Provides.

1. **EMPLOYEE (Supervises):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | F\_Name | L\_Name | Gender | Salary | DOB | Age |
| Specialization | City | Street\_Nb | Building | Floor | Supervisor \_ID |

An employee can oversee many other employees. The “Supervises” is a self-referencing relationship on the EMPLOYEE entity type. Therefore, we use the relation EMPLOYEE. Furthermore, we add to the relation the foreign key ID which is the primary key of EMPLOYEE entity type and we name it ‘*Superviser\_ID’*.

1. **EMPLOYEE (Works\_for):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | F\_Name | L\_Name | Gender | Salary | DOB | Age |
| Specialization | City | Street\_Nb | Building | Floor | Supervisor \_ID | DPT\_Nb |

Many employees work for a particular department. The "Works\_for" is a relationship between the EMPLOYEE entity type and the DEPARTMENT entity type. In this case, the "many" side is the EMPLOYEE. Therefore, we use the previously modified relation "EMPLOYEE". Furthermore, we add to the relation the foreign key number which is the primary key of DEPARTMENT entity type, and name it *‘DPT\_Nb’*.

1. **PLANT (Is\_stored):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Code | Species | Name | Price | Age | Water\_usage | BRANCH\_Nb |

Many plants are stored in one branch. The “IS\_STORED” is a relationship between the PLANT entity type and the BRANCH entity type. In this case, the “many” side is the PLANT entity type. Therefore, we use the relation "PLANT". Furthermore, we add to the relation the foreign key number which is the primary key of the BRANCH entity type, and name it *‘BRANCH\_NB’*.

1. **PLANT (Transports):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Code | Species | Name | Price | Age | Water\_usage | BRANCH\_Nb |
| VHC-S/N |

One vehicle transports many plants. The “TRANSPORTS” is a relationship between the PLANT entity type and the VEHICLE entity type. In this case, the “many” side is the PLANT entity type. Therefore, we use the previously modified relation "PLANT". Furthermore, we add to the relation the foreign key serial number which is the primary key of the VEHICLE entity type, and name it *‘VHC\_S/N’*.

1. **EMPLOYEE (Works\_in):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | F\_Name | L\_Name | Gender | Salary | DOB | Age |
| Specialization | City | Street\_Nb | Building | Floor | Supervisor \_ID | DPT\_Nb |
| GRN\_Nb |

Some employees might work in greenhouses like botanists and agricultures. Hence, the "Works\_in" is a relationship between the EMPLOYEE and GREENHOUSE entity types. In this case, the EMPLOYEE is the "many" side. Therefore, we use the previously modified relation "EMPLOYEE". Furthermore, we add to the relation the foreign key number which is the primary key of GREENHOUSE and name it *‘GRN\_Nb’*.

1. **EQUIPMENT (Needs):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S/N | Type | Cost | Status | GRN\_Nb |

One greenhouse needs several types of equipment. The "NEED" is a relationship between the GREENHOUSE entity type and the EQUIPMENT entity type. In this case, the "many" side is the EQUIPMENT entity type. Therefore, we use the relation "EQUIPMENT". Furthermore, we add to the relation the foreign key number which is the primary key of the GREENHOUSE entity type, and name it *‘GRN\_Nb’*.

1. **PLANT (Is\_located):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Code | Species | Name | Price | Age | Water\_usage | BRANCH\_Nb |
| VHC\_S/N | GRN\_Nb |

Many plants are located in one greenhouse. The “IS\_LOCATED” is a relationship between the GREENHOUSE entity type and the PLANT entity type. In this case, the “many” side is the PLANT entity type. Therefore, we use the previously modified relation “PLANT”. Furthermore, we add to the relation the foreign key number which is the primary key of the GREENHOUSE entity type, and name it *‘GRN\_Nb’*.

1. **STORAGE (Has\_a):**

|  |  |  |
| --- | --- | --- |
| Storage\_number | Maximum\_capacity | GRN\_Nb |

One greenhouse has many storages. The “HAS\_A” is a relationship between the GREENHOUSE entity type and the STORAGE entity type. In this case, the “many” side is the STORAGE entity type. Therefore, we use the relation “STORAGE”. Furthermore, we add to the relation the foreign key number which is the primary key of the GREENHOUSE entity type, and name it *‘GRN\_Nb’*.

1. **NUTRIENT (Stores):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Code | Name | Cost | Expiry\_  date | Time\_  to\_Expire | STRG\_Nb | Quantity\_  stored |

One storage stores many nutrients. The “STORES” is a relationship between the NUTRIENT entity type and the STORAGE entity type. In this case, the “many” side is the NUTRIENT entity type. Therefore, we use the relation "NUTRIENT". Furthermore, we add to the relation the foreign key number which is the primary key of the STORAGE entity type, and name it *‘STRG\_Nb’* in addition to the simple attribute of Stores *‘Quantity\_stored’*.

1. **EQUIPMENT (Contains):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S/N | Type | Cost | Status | GRN\_Nb | STRG\_Nb |

One storage contains several types of equipment. The "CONTAIN" is a relationship between the STORAGE entity type and the EQUIPMENT entity type. In this case, the "many" side is the EQUIPMENT entity type. Therefore, we use the previously modified relation "EQUIPMENT". Furthermore, we add to the relation the foreign key number which is the primary key of the STORAGE entity type, and name it *‘STRG\_Nb’*.

1. **NUTRIENT (Supplies\_1):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Code | Name | Cost | Expiry\_  date | Time\_  to\_Expire | STRG\_Nb | Quantity\_  stored |
| SUPP\_ID | Quantity\_  supplied |

One supplier supplies many nutrients. The “SUPPLIES\_1” is a relationship between the NUTRIENT entity type and the SUPPLIER entity type. In this case, the “many” side is the NUTRIENT entity type. Therefore, we use the previously modified relation "NUTRIENT". Furthermore, we add to the relation the foreign key ID which is the primary key of the SUPPLIER entity type, and name it *‘SUPP\_ID’*, in addition to the simple attribute of the relation called *‘Quantity\_supplied’*.

1. **EQUIPMENT (Supplies\_2):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S/N | Type | Cost | Status | GRN\_Nb | STRG\_Nb |
| Quantity\_  stored | SUPP\_ID |

One supplier supplies several types of equipment. The "SUPPLIES\_2" is a relationship between the EQUIPMENT entity type and the SUPPLIER entity type. In this case, the "many" side is the EQUIPMENT entity type. Therefore, we use the previously modified relation "EQUIPMENT". Furthermore, we add to the relation the foreign key ID which is the primary key of the SUPPLIER entity type, and name it *‘SUPP\_ID’*.

1. **PLANT (Provides):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Code | Species | Name | Price | Age | Water\_usage | BRANCH\_Nb |
| VHC\_S/N | GRN\_Nb | SUPP\_ID |

One supplier provides many plants. The “PROVIDES” is a relationship between the PLANT entity type and the SUPPLIER entity type. In this case, the “many” side is the PLANT entity type. Therefore, we use the previously modified relation "PLANT". Furthermore, we add to the relation the foreign key ID which is the primary key of the SUPPLIER entity type, and name it *‘SUPP\_ID’*.

## **STEP 5:** Mapping of M:N Relationship Types

In this step, we are going to map the binary many-to-many relationships. For each many-to-many relationship, we are going to create a new relationship that includes, as foreign keys, the primary keys of all participating relations. Their combination will form the primary key of this newly created relation. We must also include any other simple attribute of the many-to-many relationship. The many-to-many relationships needed to be mapped are: Drives, Purchases\_from, Utilizes, and Requires.

1. **Drives:**

|  |  |
| --- | --- |
| EMP-ID | VHC-S/N |

Many employees drive many vehicles. The “DRIVES” relationship links the EMPLOYEE entity type and the VEHICLE entity type. We create a new relation called "DRIVES" that includes the primary key of both entities EMPLOYEE and VEHICLE. The primary key of the EMPLOYEE entity type, ID, is added to this relation and renamed *‘EMP-ID’*. Also, the primary key of VEHICLE, S/N, is added and renamed *‘VHC-S/N’*. The combination of those two keys forms the primary key of this relation.

1. **PURCHASES\_FROM:**

|  |  |  |
| --- | --- | --- |
| CUST-ID | BRANCH-Nb | Quantity\_purchased |

Many customers purchase from many branches. The “PURCHASE\_FROM” relationship links the CUSTOMER entity type and the BRANCH entity type. We create a new relation called "PURCHASE\_FROM" that includes the primary key of both entities CUSTOMER and BRANCH. The primary key of the CUSTOMER entity type, ID, is added to this relation and renamed *‘CUST-ID’*. Also, the primary key of BRANCH, branch\_number, is added and renamed *‘BRANCH\_Nb’*. The combination of those two keys forms the primary key of this relation. We also add the simple attribute of this relation called *‘Quantity\_purchased’*.

1. **UTILIZES:**

|  |  |
| --- | --- |
| EMP-ID | EQUIP-S/N |

Many employees use several types of equipment. The "UTILIZES" relationship links the EMPLOYEE entity type and the EQUIPMENT entity type. We create a new relation called "UTILIZES" that includes the primary key of both entities EMPLOYEE and EQUIPMENT. The primary key of the EMPLOYEE entity type, ID, is added to this relation and renamed *‘EMP-ID’*. Also, the primary key of EQUIPMENT, S/N, is added and renamed *‘EQUIP-S/N’*. The combination of those two keys forms the primary key of this relation.

1. **REQUIRES:**

|  |  |  |
| --- | --- | --- |
| Plant-Code | Nutrient-Code | Dosage |

Many plants require many nutrients. The “REQUIRES” relationship links the PLANT entity type and the NUTRIENT entity type. We create a new relation called "REQUIRES" that includes the primary key of both entities PLANT and NUTRIENT. The primary key of the PLANT entity type, code, is added to this relation and renamed *‘Plant-Code’*. Also, the primary key of NUTRIENT, S/N, is added and renamed *‘Nutrient-Code’*. The combination of those two keys forms the primary key of this relation. We also add the simple attribute of this relation called *‘Dosage’*.

## **STEP 6:** Mapping of Multivalued Attributes

In this step, we are going to map the multivalued attributes which we ignored before. For each multivalued attribute, we create a new relation containing the related attribute and the primary key of the entity type to which it belongs. Their combination will represent the primary key of the newly created relation. We have three multivalued attributes which are: CUST\_PURCHASE\_DATES, EMP\_PHONE\_NUMBER, and SUPP\_SUPPLY\_DATES.

1. **CUST\_PURCHASE\_DATES:**

|  |  |
| --- | --- |
| CUST-ID | Purchase-dates |

The multivalued attribute Purchase-dates belong to the CUSTOMER entity type. To illustrate it, we create a relation called "CUST\_PURCHASE\_DATES". Its primary key is made of the primary key of the CUSTOMER entity type, which is *‘CUST\_ID’*, as well as the *‘Purchase-dates’* attribute that represents the multiple dates that customers have purchased on.

1. **EMP\_PHONE\_NUMBER:**

|  |  |
| --- | --- |
| EMP-ID | Phone-number |

The multivalued attribute Phone-number belongs to the EMPLOYEE entity type. To illustrate it, we create a relation called "EMP\_PHONE\_NUMBER". Its primary key is made of the primary key of the EMPLOYEE entity type, which is *‘EMP-ID’*, and the ‘*Phone-number’* attribute.

1. **SUPP\_SUPPLY\_DATES:**

|  |  |
| --- | --- |
| SUPP-ID | Supply-dates |

The multivalued attribute Supply-dates belongs to the SUPPLIER entity type. To illustrate it, we create a relation called “SUPP\_SUPPLY\_DATES”. Its primary key is made of the primary key of the SUPPLIER entity type, which is *‘SUPP-ID’*, as well as the *‘Supply-dates’* attribute to track the dates of when these supplies are being purchased.

## **STEP 7:** Mapping of N-ary Relationship Types

In this step, we are going to map the N-ary Relationship types. We should create a new relation containing the primary keys of all participating entities and any simple attributes of the relationship type. In our design we have no N-ary relationship types, so this step is not applicable here.

## **FINAL STEP:** Final Displays

**DEPARTMENT:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Department-  number | Name | Email | Phone\_  number | City | Street\_Nb | Building |
| Floor | Emp-ID |

**CUSTOMER:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Phone\_number | Name | City | Street\_Nb |

**EMPLOYEE:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | F\_Name | L\_Name | Gender | Salary | DOB | Age |
| Specialization | City | Street\_Nb | Building | Floor | Supervisor \_ID | DPT\_Nb |
| GRN\_Nb |

**SCHEDULE:**

|  |  |  |
| --- | --- | --- |
| Emp-ID | Start-time | End-time |

**BRANCH:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Branch\_number | City | Street\_number | Email | Phone\_number | Emp-ID |

**VEHICLE:**

|  |  |  |
| --- | --- | --- |
| S/N | Registration | Type |

**GREENHOUSE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Greenhouse-number | City | Street\_name | Type | Size | Number\_of\_plants |

**PLANT:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Code | Species | Name | Price | Age | Water\_usage | BRANCH\_Nb |
| VHC\_S/N | GRN\_Nb | SUPP\_ID |

**NUTRIENT:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Code | Name | Cost | Expiry\_  date | Time\_  to\_Expire | STRG\_Nb | Quantity\_  stored |
| SUPP\_ID | Quantity\_  supplied |

**STORAGE:**

|  |  |  |
| --- | --- | --- |
| Storage\_number | Maximum\_capacity | GRN\_Nb |

**EQUIPMENT:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S/N | Type | Cost | Status | GRN\_Nb | STRG\_Nb |
| SUPP\_ID |

**SUPPLIER:**

|  |  |  |
| --- | --- | --- |
| ID | Name | Phone\_number |

**Drives:**

|  |  |
| --- | --- |
| EMP-ID | VHC-S/N |

**PURCHASES\_FROM:**

|  |  |  |
| --- | --- | --- |
| CUST-ID | BRANCH-Nb | Quantity\_purchased |

**UTILIZES:**

|  |  |
| --- | --- |
| EMP-ID | EQUIP-S/N |

**REQUIRES:**

|  |  |  |
| --- | --- | --- |
| Plant-Code | Nutrient-Code | Dosage |

**CUST\_PURCHASE\_DATES:**

|  |  |
| --- | --- |
| CUST-ID | Purchase-dates |

**EMP\_PHONE\_NUMBER:**

|  |  |
| --- | --- |
| EMP-ID | Phone-  number |

**SUPP\_SUPPLY\_DATES:**

|  |  |
| --- | --- |
| SUPP-ID | Supply-dates |

# Table Structure for GoGreen Plant Nursery:

After designing the ER diagram for GoGreen plant nursery and mapping this diagram into relational database design, now it is time to start creating the concrete tables for our database on the Oracle Database Server. We will begin by creating all tables and then inserting data into them. Finally, we will execute some queries to show the value of the database in a plant nursery.

## DEPARTMENT:

CREATE TABLE DEPARTMENT

(

Department\_number INT PRIMARY KEY,

Name VARCHAR(30) NOT NULL,

Email VARCHAR(50) NOT NULL,

Phone\_number CHAR(14),

City VARCHAR(15),

Street\_Nb INT,

Building VARCHAR(30),

Floor INT,

Emp\_ID INT

);

ALTER TABLE DEPARTMENT

ADD FOREIGN KEY(EMP\_ID)

REFERENCES EMPLOYEE(ID);

## CUSTOMER:

CREATE TABLE CUSTOMER

(

ID INT PRIMARY KEY,

Phone\_Number CHAR(14),

Name VARCHAR(30),

City VARCHAR(30),

Street\_Nb INT);

## EMPLOYEE:

CREATE TABLE EMPLOYEE

(

ID INT PRIMARY KEY,

F\_Name VARCHAR(15),

L\_Name VARCHAR(15),

Gender CHAR(1),

Salary INT,

DOB DATE NOT NULL,

AGE INT,

Specialization VARCHAR(30),

City VARCHAR(30),

Street\_Nb INT,

Building VARCHAR(50),

Floor INT,

Supervisor\_ID INT,

DPT\_Nb INT,

GRN\_Nb INT,

FOREIGN KEY(Supervisor\_ID)

REFERENCES EMPLOYEE(ID),

FOREIGN KEY(DPT\_Nb)

REFERENCES DEPARTMENT(Department\_number)

);

ALTER TABLE EMPLOYEE

ADD FOREIGN KEY(GRN\_Nb)

REFERENCES GREENHOUSE(Greenhouse\_number);

## SCHEDULE:

SCHEDULE:

CREATE TABLE SCHEDULE

(

Emp\_ID INT NOT NULL,

Start\_time TIMESTAMP,

End\_time TIMESTAMP,

FOREIGN KEY(Emp\_ID)

REFERENCES EMPLOYEE(ID)

);

## BRANCH:

CREATE TABLE BRANCH

(

Branch\_number INT PRIMARY KEY,

City VARCHAR(30),

Street\_number INT,

Email VARCHAR(50),

Phone\_number CHAR(14),

Emp\_ID INT,

FOREIGN KEY(Emp\_ID)

REFERENCES EMPLOYEE(ID)

);

## VEHICLE:

CREATE TABLE VEHICLE

(

S\_N CHAR(17) PRIMARY KEY,

Registration VARCHAR(7),

Type VARCHAR(10)

);

## GREENHOUSE:

CREATE TABLE GREENHOUSE

(

Greenhouse\_number INT PRIMARY KEY,

City VARCHAR(30),

Street\_name VARCHAR(50),

Type VARCHAR(40),

Size\_of\_Greenhouse VARCHAR(10),

Number\_of\_plants INT

);

## PLANT:

CREATE TABLE PLANT

(

Code INT PRIMARY KEY,

Species VARCHAR(30),

Name VARCHAR(50),

Price INT,

Water\_usage VARCHAR(4),

BRANCH\_Nb INT,

VHC\_S\_N CHAR(17),

GRN\_Nb INT,

SUPP\_ID INT,

Age INT,

FOREIGN KEY(BRANCH\_Nb)

REFERENCES BRANCH(Branch\_number),

FOREIGN KEY(VHC\_S\_N)

REFERENCES VEHICLE(S\_N)

);

ALTER TABLE PLANT

ADD FOREIGN KEY(GRN\_Nb)

REFERENCES GREENHOUSE(Greenhouse\_number);

ALTER TABLE PLANT

ADD FOREIGN KEY(SUPP\_ID)

REFERENCES SUPPLIER(ID);

## SUPPLIER:

CREATE TABLE SUPPLIER

(

ID INT PRIMARY KEY,

Name VARCHAR(30),

Phone\_number CHAR(13)

);

## STORAGE:

CREATE TABLE STORAGE

(

Storage\_number INT PRIMARY KEY,

Maximum\_capacity INT,

GRN\_Nb INT,

FOREIGN KEY(GRN\_Nb)

REFERENCES GREENHOUSE(Greenhouse\_number));

## NUTRIENT:

CREATE TABLE NUTRIENT

(

Code INT PRIMARY KEY,

Name VARCHAR(15),

Cost INT,

Expiry\_date DATE,

Time\_to\_Expire INT,

STRG\_Nb INT,

Quantity\_Stored INT,

SUPP\_ID INT,

Quantity\_supplied INT,

FOREIGN KEY(STRG\_Nb)

REFERENCES STORAGE(Storage\_number),

FOREIGN KEY(SUPP\_ID)

REFERENCES SUPPLIER(ID)

);

## EQUIPEMENT:

CREATE TABLE EQUIPMENT

(

S\_N VARCHAR(15) PRIMARY KEY,

Type VARCHAR(20),

Cost INT,

Status VARCHAR(20),

GRN\_Nb INT,

STRG\_Nb INT,

SUPP\_ID INT,

FOREIGN KEY (GRN\_Nb)

REFERENCES GREENHOUSE(Greenhouse\_number),

FOREIGN KEY(STRG\_Nb)

REFRENCES STORAGE(Storage\_number),

FOREIGN KEY(SUPP\_ID)

REFERENCES SUPPLIER(ID)

);

## DRIVES:

CREATE TABLE DRIVES

(

EMP\_ID INT NOT NULL,

VHC\_S\_N CHAR(17) NOT NULL,

FOREIGN KEY(EMP\_ID)

REFERENCES EMPLOYEE(ID),

FOREIGN KEY(VHC\_S\_N)

REFERENCES VEHICLE(S\_N)

);

## PURCHASES\_FROM:

CREATE TABLE PURCHASES\_FROM

(

CUST\_ID INT,

BRANCH\_Nb INT,

Quantity\_purchased INT,

FOREIGN KEY(CUST\_ID)

REFERENCES CUSTOMER(ID),

FOREIGN KEY(BRANCH\_Nb)

REFERENCES BRANCH(Branch\_number)

);

## UTILIZES:

CREATE TABLE UTILIZES

(

EMP\_ID INT,

EQUIP\_S\_N VARCHAR(17),

FOREIGN KEY(EMP\_ID)

REFERENCES EMPLOYEE(ID),

FOREIGN KEY(EQUIP\_S\_N)

REFERENCES EQUIPMENT(S\_N)

);

## REQUIRES:

CREATE TABLE REQUIRES

(

Plant\_Code INT,

Nutrient\_Code INT,

Dosage VARCHAR(10),

FOREIGN KEY(Plant\_Code)

REFERENCES PLANT(Code),

FOREIGN KEY(Nutrient\_Code)

REFERENCES NUTRIENT(Code));

## CUST\_PURCHASE\_DATES:

CREATE TABLE SUPP\_SUPPLY\_DATES

(

SUPP\_ID INT,

Supply\_dates DATE,

FOREIGN KEY(SUPP\_ID)

REFERENCES SUPPLIER (ID)

);

## EMP\_PHONE\_NUMBER:

CREATE TABLE EMP\_PHONE\_NUMBER

(

EMP\_ID INT,

Phone\_number CHAR(14),

FOREIGN KEY(EMP\_ID)

REFERENCES EMPLOYEE(ID)

);

## SUPP\_SUPPLY\_DATES:

CREATE TABLE SUPP\_SUPPLY\_DATES

(

SUPP\_ID INT,

Supply\_dates DATE PRIMARY KEY,

FOREIGN KEY(SUPP\_ID)

REFERENCES SUPPLIER (ID)

);

# Table Description:

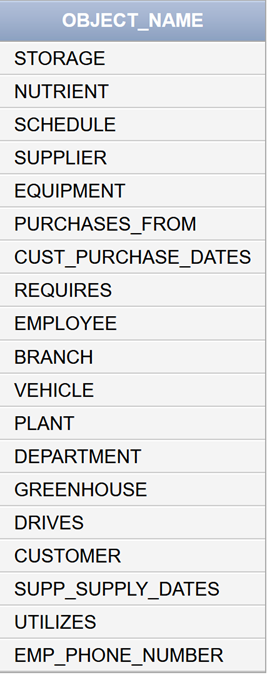
After creating all the tables on the oracle database server, we can now view the description of each table in order to make sure everything is working well, and no mistakes were made during the creation of tables.

In our database we have the following tables created on the oracle database server:

SQL> SELECT DISTINCT OBJECT\_NAME

FROM USER\_OBJECTS

WHERE OBJECT\_TYPE='TABLE';



## DEPARTMENT:

SQL> DESC DEPARTMENT;

Graphical user interface

Description automatically generated with medium confidence

## CUSTOMER:

SQL> DESC CUSTOMER;

Table

Description automatically generated

## EMPLOYEE:

SQL> DESC EMPLOYEE;

Table, calendar

Description automatically generated

## SCHEDULE:

SQL> DESC SCHEDULE;

Table

Description automatically generated

## BRANCH:

SQL> DESC BRANCH;

Table

Description automatically generated

## VEHICLE:

SQL> DESC VEHICLE;

A screenshot of a computer

Description automatically generated with medium confidence

## GREENHOUSE:

SQL> DESC GREENHOUSE;

Table

Description automatically generated

## PLANT:

SQL> DESC PLANT;

Table

Description automatically generated

## SUPPLIER:

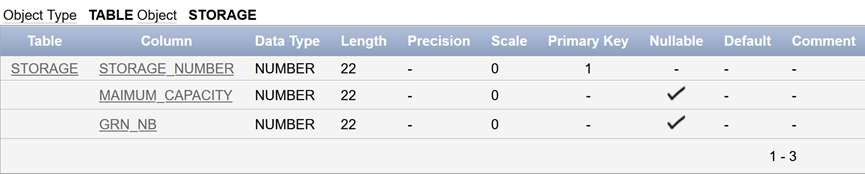
SQL> DESC SUPPLIER;

A screenshot of a computer

Description automatically generated

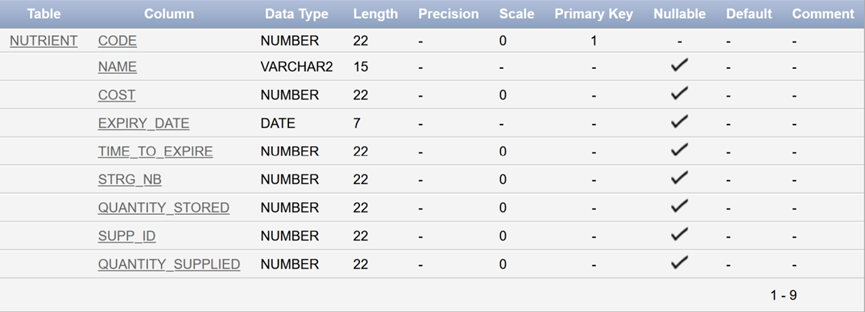
## STORAGE:

SQL> DESC STORAGE;



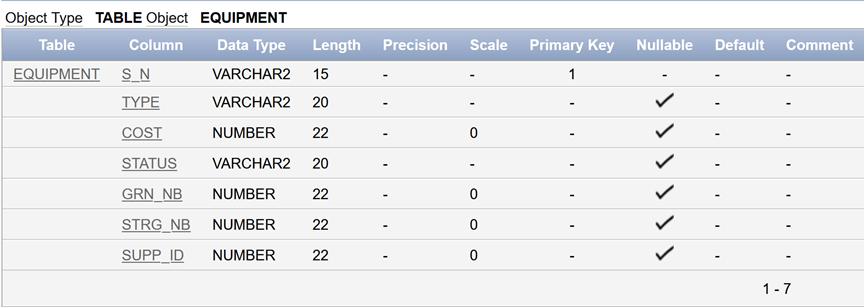
## NUTRIENT:

SQL> DESC NUTRIENT;



## EQUIPMENT:

SQL> DESC EQUIPMENT;



## DRIVES:

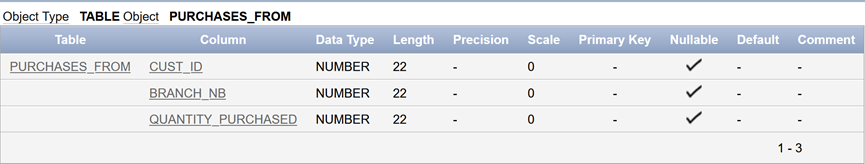
SQL> DESC DRIVES;

A screenshot of a computer

Description automatically generated with medium confidence

## PURCHASES\_FROM:

SQL> DESC PURCHASES\_FROM;



## UTILIZES:

SQL> DESC UTILIZES;

Graphical user interface, application

Description automatically generated

## REQUIRES:

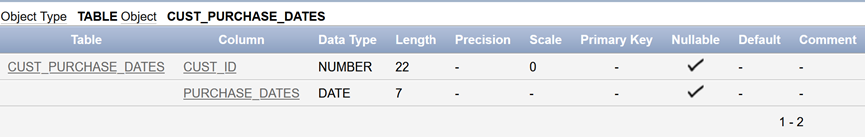
SQL> DESC REQUIRES;

Graphical user interface, application

Description automatically generated

## CUST\_PURCHASE\_DATES:

SQL> DESC CUST\_PURCHASE\_DATES;



## EMP\_PHONE\_NUMBER:

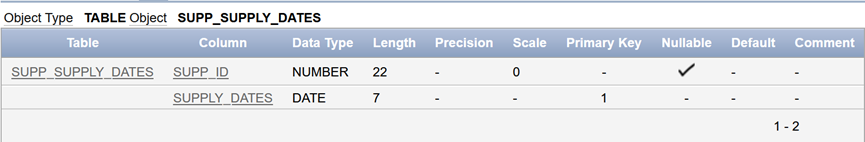
SQL> DESC EMP\_PHONE\_NUMBER;

Graphical user interface, application

Description automatically generated

## SUPP\_SUPPLY\_DATES:

SQL> DESC SUPPLY\_DATES;



# Inserting data:

## DEPARTMENT:

INSERT INTO DEPARTMENT VALUES (11, ‘Agriculture’, ‘agriculture@gogreen.co’, ‘+961 01-333444’, ‘Beirut’, 21, ‘Safadi Building’, 5, 1845905447);

INSERT INTO DEPARTMENT VALUES (12, ‘Sales’, ‘sales@gogreen.co’, ‘+961 01-222555’, ‘Beirut’, 21, ‘Nicola Building’, 3, 1122334455);

INSERT INTO DEPARTMENT VALUES (13, ‘HR’, ‘hr@gogreen.co’, ‘+961 01-955777’, ‘Beirut’, 21, ‘Go Nature Building’, 1, 1234512345);

INSERT INTO DEPARTMENT VALUES (14, ‘Research’, ‘sciresearch@gogreen.co’, ‘+961 01-433366’, ‘Beirut’, 21, ‘Beirut local building’, 2, null);

INSERT INTO DEPARTMENT VALUES (15, ‘Finance’, ‘finance@gogreen.co’, ‘+961 01-777234’, ‘Beirut’, 21, ‘Nicola Building’, 2, 5544332211);

INSERT INTO DEPARTMENT VALUES (16, ‘Nursery’, ‘nurserydepartment@gogreen.co’, ‘+961 01-297198’, ‘Beirut’, 21, ‘Beirut local building’, 1, 2307411820);

INSERT INTO DEPARTMENT VALUES (17, ‘Customer Service’, ‘customerservice@gogreen.co’, ‘+961 01-589238’, ‘Beirut’, 21 ‘Safadi Building’, 2, 4892211133);

INSERT INTO DEPARTMENT VALUES (18, ‘Branches Management’, ‘branchesmanagement@gogreen.co’, ‘+961 01-499219’, ‘Beirut’, 21, ‘Managerial Building’, 2, null);

INSERT INTO DEPARTMENT VALUES (19, ‘Vehicle Management’, ‘transportationdepartment@gogreen.co’, ‘+961 01-398138’, ‘Beirut’, 21, ‘Managerial Building’, 3, null);

INSERT INTO DEPARTMENT VALUES (20, ‘Supplies Management’, ‘suppliesdepartment@gogreen.co’, ‘+961 81-483012’, ‘Beirut’, 21, ‘Managerial Building’, 4, 5431254321);

## CUSTOMER:

INSERT INTO CUSTOMER VALUES (58147405, ‘+961 81-483924’, ‘Leyla Thana’, ‘Beirut’, 23);

INSERT INTO CUSTOMER VALUES (84397160, ‘+961 81-483308’, ‘Naziha Adila’, ‘Beqaa’, 43);

INSERT INTO CUSTOMER VALUES (95747451, ‘+961 81-293249’, ‘Samia Salman’, ‘Beqaa’, 34);

INSERT INTO CUSTOMER VALUES (61996164, ‘+961 81-492209’, ‘Anwar Shahd’, ‘Beirut’, 23);

INSERT INTO CUSTOMER VALUES (‘17664996’, ‘+961 81-394398’, ‘Talib Amina’, ‘Jnoub’, 23);

INSERT INTO CUSTOMER VALUES (42284032, ‘+961 81-129328’, ‘Rafiq Jalil’, ‘Beirut’, 23);

INSERT INTO CUSTOMER VALUES (94405296, ‘+961 81-437792’, ‘Salim Safaa’, ‘Beirut’, 53);

INSERT INTO CUSTOMER VALUES (58019557, ‘+961 81-329249’, ‘Dalia Noura’, ‘Beirut’, 51);

INSERT INTO CUSTOMER VALUES (63679144, ‘+961 81-393923’, ‘Fatin Amjad’, ‘Beirut’, 25);

INSERT INTO CUSTOMER VALUES (97303832, ‘+961 81-329293’, ‘Yasmeen Anwar’, ‘Jbeil’, 72);

INSERT INTO CUSTOMER VALUES (202, NULL, ‘Ramzi Harati’, ‘Hamra’, NULL);

## EMPLOYEE:

INSERT INTO EMPLOYEE VALUES (6784705218, ‘Latif’, ‘Ghadir’, ‘M’, 500, ‘12-01-1990’, 31, ‘Greenhouse staff’, ‘Beirut’, 21, ‘Red Ribbon’, 1, 1845905447, 11, 31);

INSERT INTO EMPLOYEE VALUES (4349797788, ‘Mariam’, ‘Wasim’, ‘F’, 1000, ’01-01-1989’, 32, ‘Branch manager’, ‘Jbeil’, 72, ‘Aqua Building’, 2, 6784705218, 18, null);

INSERT INTO EMPLOYEE VALUES (5409979193, ‘Marwa’, ‘Noura’, ‘F’, 1000, ’03-07-1988’, 33, ‘Branch manager’, ‘Beirut’, 26, ‘Smith Building’, 1, null, 18, null);

INSERT INTO EMPLOYEE VALUES (6963258256, ‘Sani’, ‘Naaji’, ‘M’, 2000, ’09-10-1972’, 49, ‘Botanist’, ‘Beirut’, 21, ‘Erwin building’, 1, 1845905447, 11, 33);

INSERT INTO EMPLOYEE VALUES (1117618988, ‘Jaffar’, ‘Sanaa’, ‘F’, 3500, ‘12-05-1970’, 51, ‘Specialized Botanist’, ‘Jbeil’, 21, ‘Younes building’, 1, 1845905447, 11, 35);

INSERT INTO EMPLOYEE VALUES (2074524800, ‘Esmail’, ‘Naqi’, ‘M’, 300, ‘02-10-1999’, 22, ‘Customer service’, ‘Jbeil’, 82, ‘Nasser building’, 2, 4892211133, 17, null);

INSERT INTO EMPLOYEE VALUES (1845905447, ‘Safa’, ‘Gabr’, ‘F’, 5000, ‘12-12-1978’, 43, ‘Supervisor’, ‘Beirut’, 21, ‘Wafic Building’, 1, null, 11, null);

INSERT INTO EMPLOYEE VALUES (2307411820, ‘Samad’, ‘Faruq’, ‘M’, 5500, ‘12-06-1976’, 45, ‘Supervisor’, ‘Khalde’, ‘Near Choifeit Street’, ‘Wafic Building’,1, null, 16, null);

INSERT INTO EMPLOYEE VALUES (7487344706, ‘Danial’, ‘Maryam’, ‘M’, 4000, ‘12-07-1995’, 26, ‘Greenhouse staff’, ‘Beirut’, 21, ‘Dana Building’, 3, 1845905447, 11, 34);

INSERT INTO EMPLOYEE VALUES (8610870297, ‘Ahmad’, ‘Mukhtar’, ‘M’, 3900, ‘12-11-1996’, 25, ‘Greenhouse staff’, ‘Beirut’, 29, ‘Dana Building’, 4, 1845905447, 11, 34);

INSERT INTO EMPLOYEE VALUES (4892211133, ‘Marian’, ‘Kaouk’, ‘F’, 5000, ‘09-12-1978’, 43, ‘Supervisor’, ‘Beirut’, 23, ‘Younes Building’, 1, null, 17, null);

INSERT INTO EMPLOYEE VALUES (1122334455, ‘Farah’, ‘Bizri’, ‘F’, 5000, ‘11-12-1978’, 43, ‘Supervisor’, ‘Khalde’, 27, ‘Khaddage Building’, 1, null, 12, null);

INSERT INTO EMPLOYEE VALUES (1234512345, ‘Paul’, ‘Maroun’, ‘M’, 5100, ‘11-12-1978’, 43, ‘Supervisor’, ‘Hamra’, 14, ‘Smeha Building’, 1, null, 13, null);

INSERT INTO EMPLOYEE VALUES (5544332211, ‘Ali’, ‘Mawla’, ‘M’, 5100, ‘11-11-1979’, 42, ‘Supervisor’, ‘Jbeil’, 88, ‘Kwizatz Building’, 5, null, 15, null);

INSERT INTO EMPLOYEE VALUES (5431254321, ‘Moustapha’, ‘Nasser’, ‘M’, 5100, ‘11-11-1981’, 40, ‘Supervisor’, ‘Jbeil’, 89, ‘Hazeratch Building’, 3, null, 20, null);

## SCHEDULE:

INSERT INTO SCHEDULE VALUES (6784705218, TIMESTAMP ‘2019-12-1 09:00:00’, TIMESTAMP ‘2020-12-1 17:00:00’);

INSERT INTO SCHEDULE VALUES (4349797788, TIMESTAMP ‘2020-12-1 10:00:00’, TIMESTAMP ‘2021-12-1 17:00:00’);

INSERT INTO SCHEDULE VALUES (5409979193, TIMESTAMP ‘2019-12-1 09:00:00’, TIMESTAMP ‘2020-12-1 18:00:00’);

INSERT INTO SCHEDULE VALUES (6963258256, TIMESTAMP ‘2019-12-1 08:00:00’, TIMESTAMP ‘2020-12-1 17:00:00’);

INSERT INTO SCHEDULE VALUES (1117618988, TIMESTAMP ‘2019-12-1 09:00:00’, TIMESTAMP ‘2020-12-1 17:00:00’);

INSERT INTO SCHEDULE VALUES (2074524800, TIMESTAMP ‘2019-12-1 09:00:00’, TIMESTAMP ‘2020-12-1 18:00:00’);

INSERT INTO SCHEDULE VALUES (1845905447, TIMESTAMP ‘2020-12-1 09:00:00’, TIMESTAMP ‘2021-12-1 17:00:00’);

INSERT INTO SCHEDULE VALUES (2307411820, TIMESTAMP ‘2020-12-1 09:00:00’, TIMESTAMP ‘2021-12-1 19:00:00’);

INSERT INTO SCHEDULE VALUES (7487344706, TIMESTAMP ‘2020-12-1 11:00:00’, TIMESTAMP ‘2021-12-1 19:00:00’);

INSERT INTO SCHEDULE VALUES (8610870297, TIMESTAMP ‘2019-12-1 11:00:00’, TIMESTAMP ‘2020-12-1 19:00:00’);

## BRANCH:

INSERT INTO BRANCH VALUES (21, ‘Beirut Hamra’, 13, ‘gogreenbeiruthamra@gogreen.co’, ‘+961 81-937298’, 6784705218);

INSERT INTO BRANCH VALUES (22, ‘Jbeil’, 81, ‘gogreenjbeil@gogreen.co’, ‘+961 81-4929439’, 2074524800);

INSERT INTO BRANCH VALUES (23, ‘Baalbek’, 18, ‘gogreenbaalbek@gogreen.co’, ‘+961 81-935941’, 6784705218);

INSERT INTO BRANCH VALUES (24, ‘Beqaa’, 21, ‘gogreenbeqaa@gogreen.co’, ‘+961 81-483484’, 2074524800);

INSERT INTO BRANCH VALUES (25, ‘Beirut Verdun’, 73, ‘gogreenbeirutverdun@gogreen.co’, ‘+961 81-494942’, 6784705218);

INSERT INTO BRANCH VALUES (26, ‘Beirut Gemeiyze’, 32, ‘gogreenbeirutgemeiyze@gogreen.co’, ‘+961 81-289394’, 2074524800);

INSERT INTO BRANCH VALUES (27, ‘Tyre’, 41, ‘gogreentyre@gogreen.co’, ‘+961 81-923937’, 2307411820);

INSERT INTO BRANCH VALUES (28, ‘Tripoli’, 63, ‘gogreentrablous@gogreen.co’, ‘+961 81-148937’, 2307411820);

INSERT INTO BRANCH VALUES (29, ‘Aley’, 79, ‘gogreenaley@gogreen.co’, ‘+961 81-652837’, 7487344706);

INSERT INTO BRANCH VALUES (30, ‘Beirut Mar Mikhael’, 55, ‘gogreenbeirutmarmakhiel@gogreen.co’, ‘+961 76-575180’, 7487344706);

## VEHICLE:

INSERT INTO VEHICLE VALUES (‘1N4BL2AP8BN503925’, ‘M392218’, ‘Nissan’);

INSERT INTO VEHICLE VALUES (‘KNDJN2A27E7740516’, ‘M113179’, ‘Ford’);

INSERT INTO VEHICLE VALUES (‘JTJZK1BA8A2403790’, ‘A123456’, ‘Nissan’);

INSERT INTO VEHICLE VALUES (‘1GKET16S426107309’, ‘B682683’, ‘Toyota’);

INSERT INTO VEHICLE VALUES (‘5NPE34AF8FH082224’, ‘B428473’, ‘Hunda’);

INSERT INTO VEHICLE VALUES (‘2HNYD18753H504973’, ‘G385580’, ‘Audi’);

INSERT INTO VEHICLE VALUES (‘4V4NC9TK36N430345’, ‘O168579’, ‘Hunda’);

INSERT INTO VEHICLE VALUES (‘2G1WX15K429306075’, ‘B459078’, ‘Hunda’);

INSERT INTO VEHICLE VALUES (‘19UUA66274A046775’, ‘M422982’, ’Nissan’);

INSERT INTO VEHICLE VALUES (‘3C3CFFBR3CT340684’, ‘G100404’, ’Hunda’);

## GREENHOUSE:

INSERT INTO GREENHOUSE VALUES (31, ‘Beirut’, ‘Hamra street’, ‘Freestanding’, ‘250m^2’, 3000);

INSERT INTO GREENHOUSE VALUES (32, ‘Jbeil’, ‘Byblos street’, ‘Freestanding’, ‘300m^2’, 3500);

INSERT INTO GREENHOUSE VALUES (33, ‘Beqaa’, ‘Chtoura street’, ‘Freestanding’, ‘400m^2’, 4000);

INSERT INTO GREENHOUSE VALUES (34, ‘Beirut’, ‘Verdun street’, ‘Attached’, ‘150m^2’, 2000);

INSERT INTO GREENHOUSE VALUES (35, ‘Jbeil’, ‘Mastita street’, ‘Freestanding’, ‘600m^2’, 6000);

INSERT INTO GREENHOUSE VALUES (36, ‘Aley’, ‘Bsous street’, ‘Attached’, ‘200m^2’, 2500);

INSERT INTO GREENHOUSE VALUES (37, ‘Tyre’, ‘El Kouds street’, ‘Attached’, ‘250m^2’, 3000);

INSERT INTO GREENHOUSE VALUES (38, ‘Tripoli’, ‘El Thakafa street’, ‘Freestanding’, ‘600m^2’, 6000);

INSERT INTO GREENHOUSE VALUES (39, ‘Faraiya’, ‘Faraya street’, ‘Attached’, ‘250m^2’, 3000);

INSERT INTO GREENHOUSE VALUES (40, ‘Kfardebian’, ‘Mazraat Kfardibiane’, ‘Freestanding’, ‘400m^2’, 4000);

## PLANT:

INSERT INTO PLANT VALUES (9091, ‘Acer’, ‘amplum’, 60, ‘3%’, 21, ‘1N4BL2AP8BN503925’, 31, 313369229, 2);

INSERT INTO PLANT VALUES (9100, ‘Acer’, ‘barbinerve’, 50, ‘2%’, 22, ‘KNDJN2A27E7740516’, 32, 647010725, 1);

INSERT INTO PLANT VALUES (9119, ‘Narcissus’, ‘dubius’, 34, ‘3%’, 23, ‘JTJZK1BA8A2403790’, 33, 633281215, 3);

INSERT INTO PLANT VALUES (9491, ‘Acer’, ‘argutum’, 90, ‘4%’, 24, ‘1GKET16S426107309’, 34, 716552266, 5);

INSERT INTO PLANT VALUES (9161, ‘Narcissus’, ‘tazetta’, 20, ‘1%’, 25, ‘5NPE34AF8FH082224’, 35, 653634641, 1);

INSERT INTO PLANT VALUES (9661, ‘Bamboo’, ‘acidosasa’, 88, ‘3%’, 26, ‘2HNYD18753H504973’, 36, 636446549, 3);

INSERT INTO PLANT VALUES (6996, ‘Oxalis’, ‘alpina’, 19, ‘1%’, 27, ‘4V4NC9TK36N430345’, 37, 653464364, 1);

INSERT INTO PLANT VALUES (6975, ‘Oxalis’, ‘alata’, 22, ‘4%’, 28, ‘2G1WX15K429306075’, 38, 128373291, 1);

INSERT INTO PLANT VALUES (6361, ‘Oxalis’, ‘acetosella’, 24, ‘1%’, 29, ‘19UUA66274A046775’, 39, 238237842, 2);

INSERT INTO PLANT VALUES (6496, ‘Dahlia’, ‘excelsa’, 69, ‘5%’, 30, ‘3C3CFFBR3CT340684’, 40, 283282497, 3);

INSERT INTO PLANT VALUES (6493, ‘Dahlia’, ‘imperialis’, 63, ‘5%’, 21, ‘1N4BL2AP8BN503925’, 31, 313369229, 4);

INSERT INTO PLANT VALUES (8655, ‘Ficus’, ‘abelii’, 69, ‘5%’, 22, ‘KNDJN2A27E7740516’, 32, 647010725, 3);

INSERT INTO PLANT VALUES (1870, ‘Ficus’, ‘adelpha’, 63, ‘5%’, 23, ‘JTJZK1BA8A2403790’, 33, 633281215, 3);

INSERT INTO PLANT VALUES (9861, ‘Ficus’, ‘lecardii’, 68, ‘5%’, 24, ‘1GKET16S426107309’, 34, 716552266, 3);

INSERT INTO PLANT VALUES (6936, ‘salix’, ‘acutifolia’, 62, ‘5%’, 25, ‘5NPE34AF8FH082224’, 35, 653634641, 3);

## Storage:

INSERT INTO STORAGE VALUES (41,3500,31);

INSERT INTO STORAGE VALUES (42,4000,32);

INSERT INTO STORAGE VALUES (43,5000,33);

INSERT INTO STORAGE VALUES (44,3000,34);

INSERT INTO STORAGE VALUES (45,7000,35);

INSERT INTO STORAGE VALUES (46,3000,36);

INSERT INTO STORAGE VALUES (47,4000,37);

INSERT INTO STORAGE VALUES (48,6500,31);

INSERT INTO STORAGE VALUES (49,5050,39);

INSERT INTO STORAGE VALUES (50,5000,40);

## Nutrient:

INSERT INTO NUTRIENT VALUES (2121, ‘Iron’, 913, ‘10/10/2030’, 9, 41, 200, 313369229, 100);

INSERT INTO NUTRIENT VALUES (2122, ‘Manganese’, 513, ‘12/20/2022’, 1, 42, 200, 647010725, 300);

INSERT INTO NUTRIENT VALUES (2123, ‘Boron’, 303, ‘12/12/2024’, 3, 43, 200, 633281215,300);

INSERT INTO NUTRIENT VALUES (2124, ‘Chloride’, 213, ‘11/11/2023’, 2, 44, 500, 716552266, 500);

INSERT INTO NUTRIENT VALUES (2126, ‘Calcium’, 213, ‘12/30/2023’, 2, 46, 200, 636446549, 300);

INSERT INTO NUTRIENT VALUES (2127, ‘Potassium’, 503, ‘12/31/2024’, 3, 47, 500, 653464364, 500);

INSERT INTO NUTRIENT VALUES (2128, ‘Sulfur’, 273, ‘1/1/2022’, 1, 48, 100, 128373291, 400);

INSERT INTO NUTRIENT VALUES (2129, ‘Nitrogen’, 214, ‘10/20/2023’, 2, 49, 1025, 238237842, 1025);

INSERT INTO NUTRIENT VALUES (2130, ‘Magnesium’, 55, ‘2/2/2025’, 4, 50, 500, 283282497, 500);

INSERT INTO NUTRIENT VALUES (2131, ‘Zinc’, 213, ‘12/12/2020’, 0, 42, 100, 313369229, 100);

## Supplier:

INSERT INTO SUPPLIER VALUES (313369229, ‘Ruya Taliba’, ‘+961 71884141’);

INSERT INTO SUPPLIER VALUES (647010725, ‘Joe Helo’, ‘+961 81743718’);

INSERT INTO SUPPLIER VALUES (633281215, ‘Hashem Safadi’, ‘+961 81324032’);

INSERT INTO SUPPLIER VALUES (716552266, ‘Ibrahim Arayssi’, ‘+961 76324232’);

INSERT INTO SUPPLIER VALUES (653634641, ‘Firas Jabar’, ‘+961 70324132’);

INSERT INTO SUPPLIER VALUES (636446549, ‘Ellyas Mahfoos’, ‘+961 81689689’);

INSERT INTO SUPPLIER VALUES (653464364, ‘Genevieve Khoury’, ‘+961 76131611’);

INSERT INTO SUPPLIER VALUES (128373291, ‘Jomana Abo Al Hosn’, ‘+961 81323232’);

INSERT INTO SUPPLIER VALUES (238237842, ‘Iman Sleiman’, ‘+961 91424032’);

INSERT INTO SUPPLIER VALUES (283282497, ‘Ali Hareb’, ‘+961 71324245’);

## Equipment:

INSERT INTO EQUIPMENT VALUES (‘ABC872902072866’, ‘Pruner’, 100, ‘Used’, 31, 48, 313369229);

INSERT INTO EQUIPMENT VALUES (‘FEA819837167660’, ‘Glimour\_thumb’, 200, ‘Used’, 31, 41, 633281215);

INSERT INTO EQUIPMENT VALUES (‘ABC255829920279’, ‘FC50A\_ferticart’, 100, ‘Used’, 34, 44, 313369229);

INSERT INTO EQUIPMENT VALUES (‘FEA574455109197’, ‘gallon\_container’, 20, ‘Available’, 36, 46, 128373291);

INSERT INTO EQUIPMENT VALUES (‘ABC445294540734’, ‘hoe’, 40, ‘Available’, 31, 41, 128373291);

INSERT INTO EQUIPMENT VALUES (‘KEG943070065519’, ‘Shovel’, 45, ‘Available’, 31, 48, 313369229);

INSERT INTO EQUIPMENT VALUES (‘HRC770764478370’, ‘Axes’,45, ‘Available’, 34, 44,283282497);

INSERT INTO EQUIPMENT VALUES (‘HRC194778195255’, ‘Trowels’, 45, ‘Available’, 31, 41,128373291);

INSERT INTO EQUIPMENT VALUES (‘KEG577302200145’, ‘cloches’, 40, ‘Available’, 34, 44, 283282497);

INSERT INTO EQUIPMENT VALUES (‘DVN418938152080’, ‘Rake’, 40, ‘Available’, 39, 49, 128373291);

## Drives:

INSERT INTO DRIVES VALUES (1845905447, ‘1N4BL2AP8BN503925’);

INSERT INTO DRIVES VALUES (5409979193, ‘3C3CFFBR3CT340684’);

INSERT INTO DRIVES VALUES (1845905447, ‘4V4NC9TK36N430345’);

INSERT INTO DRIVES VALUES (5409979193, ‘1N4BL2AP8BN503925’);

INSERT INTO DRIVES VALUES (1845905447, ‘JTJZK1BA8A2403790’);

INSERT INTO DRIVES VALUES (5409979193, ‘4V4NC9TK36N430345’);

INSERT INTO DRIVES VALUES (2074524800, ‘JTJZK1BA8A2403790’);

INSERT INTO DRIVES VALUES (2074524800, ‘3C3CFFBR3CT340684’);

INSERT INTO DRIVES VALUES (2074524800, ‘2HNYD18753H504973’);

INSERT INTO DRIVES VALUES (2074524800, ‘1N4BL2AP8BN503925’);

## PURCHASES\_FROM

INSERT INTO PURCHASES\_FROM VALUES (58147405, 21, 2);

INSERT INTO PURCHASES\_FROM VALUES (84397160, 22, 1);

INSERT INTO PURCHASES\_FROM VALUES (95747451, 23, 3);

INSERT INTO PURCHASES\_FROM VALUES (61996164, 24, 4);

INSERT INTO PURCHASES\_FROM VALUES (17664996, 24, 1);

INSERT INTO PURCHASES\_FROM VALUES (42284032, 25, 2);

INSERT INTO PURCHASES\_FROM VALUES (94405296, 25, 5);

INSERT INTO PURCHASES\_FROM VALUES (58019557, 25, 1);

INSERT INTO PURCHASES\_FROM VALUES (63679144, 28, 3);

INSERT INTO PURCHASES\_FROM VALUES (97303832, 29, 7);

## UTILIZES

INSERT INTO UTILIZES VALUES (6784705218, ‘ABC872902072866’);

INSERT INTO UTILIZES VALUES (6784705218, ‘FEA819837167660’);

INSERT INTO UTILIZES VALUES (6784705218, ‘KEG943070065519’);

INSERT INTO UTILIZES VALUES (6784705218, ‘ABC445294540734’);

INSERT INTO UTILIZES VALUES (7487344706, ‘ABC255829920279’);

INSERT INTO UTILIZES VALUES (7487344706, ‘HRC770764478370’);

INSERT INTO UTILIZES VALUES (7487344706, ‘KEG577302200145’);

INSERT INTO UTILIZES VALUES (8610870297, ‘ABC255829920279’);

INSERT INTO UTILIZES VALUES (8610870297, ‘KEG577302200145’);

INSERT INTO UTILIZES VALUES (8610870297, ‘HRC770764478370’);

## REQUIRES:

INSERT INTO REQUIRES VALUES (9091, 2121, ‘100mg’);

INSERT INTO REQUIRES VALUES (9100, 2122, ‘130mg’);

INSERT INTO REQUIRES VALUES (6996, 2127, ‘123mg’);

INSERT INTO REQUIRES VALUES (9091, 2128, ‘90mg’);

INSERT INTO REQUIRES VALUES (6496, 2130, ‘80mg’);

INSERT INTO REQUIRES VALUES (8655, 2122, ‘120mg’);

INSERT INTO REQUIRES VALUES (9100, 2131, ‘110mg’);

INSERT INTO REQUIRES VALUES (9861, 2124, ‘130mg’);

INSERT INTO REQUIRES VALUES (6493, 2121, ‘100mg’);

INSERT INTO REQUIRES VALUES (6493, 2128, ‘80mg’);

## CUST\_PURCHASE\_DATES:

INSERT INTO CUST\_PURCHASE\_DATES VALUES (58147405, ‘07-11-2021’);

INSERT INTO CUST\_PURCHASE\_DATES VALUES (84397160, ‘09-17-2021’);

INSERT INTO CUST\_PURCHASE\_DATES VALUES (95747451, ‘09-17-2021’);

INSERT INTO CUST\_PURCHASE\_DATES VALUES (61996164, ‘09-20-2021’);

INSERT INTO CUST\_PURCHASE\_DATES VALUES (17664996, ‘09-27-2021’);

INSERT INTO CUST\_PURCHASE\_DATES VALUES (42284032, ‘03-08-2021’);

INSERT INTO CUST\_PURCHASE\_DATES VALUES (94405296, ‘08-17-2021’);

INSERT INTO CUST\_PURCHASE\_DATES VALUES (58019557, ‘09-04-2021’);

INSERT INTO CUST\_PURCHASE\_DATES VALUES (63679144, ‘07-21-2021’);

INSERT INTO CUST\_PURCHASE\_DATES VALUES (97303832, ‘08-28-2021’);

## EMP\_PHONE\_NUMBER:

INSERT INTO EMP\_PHONE\_NUMBER VALUES (6784705218, ‘+961 76-575180’);

INSERT INTO EMP\_PHONE\_NUMBER VALUES (4349797788, ‘+961 09-795922’);

INSERT INTO EMP\_PHONE\_NUMBER VALUES (5409979193, ‘+961 76-257312’);

INSERT INTO EMP\_PHONE\_NUMBER VALUES (6963258256, ‘+961 09-159632’);

INSERT INTO EMP\_PHONE\_NUMBER VALUES (1117618988, ‘+961 03-886442’);

INSERT INTO EMP\_PHONE\_NUMBER VALUES (2074524800, ‘+961 03-987543’);

INSERT INTO EMP\_PHONE\_NUMBER VALUES (1845905447, ‘+961 76-014532’);

INSERT INTO EMP\_PHONE\_NUMBER VALUES (2307411820, ‘+961 09-358324’);

INSERT INTO EMP\_PHONE\_NUMBER VALUES (7487344706, ‘+961 76-575191’);

INSERT INTO EMP\_PHONE\_NUMBER VALUES (8610870297, ‘+961 76-757919’);

## SUPP\_SUPPLY\_DATES:

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (313369229, ‘07-11-2021’);

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (313369229, ‘04-11-2021’);

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (633281215, ‘22-09-2020’);

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (313369229, ‘19-03-2020’);

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (653464364, ‘21-06-2019’);

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (633281215, ‘10-10-2020’);

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (653464364, ‘11-11-2019’);

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (283282497, ‘05-12-2020’);

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (283282497, ‘15-01-2021’);

INSERT INTO SUPP\_SUPPLY\_DATES (SUPP\_ID, SUPPLY\_DATES) VALUES (283282497, ‘13-04-2021’);

# Final Tables State:

## DEPARTMENT:

Graphical user interface

Description automatically generated

## CUSTOMER:

Table

Description automatically generated

## EMPLOYEE:

Table

Description automatically generated

## SCHEDULE:

Table

Description automatically generated

## BRANCH:

Table

Description automatically generated

## VEHICLE:

Table

Description automatically generated

## GREENHOUSE:

Table

Description automatically generated

## PLANT:

Table

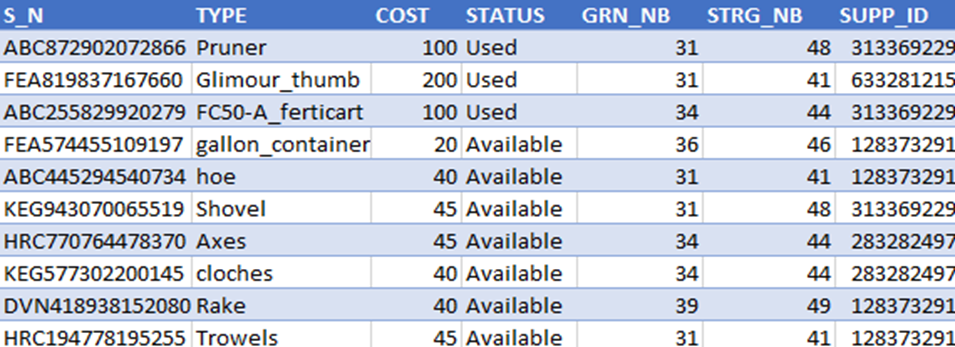
Description automatically generated

## STORAGE:

Table

Description automatically generated

## EQUIPMENT:



## PURCHASED\_FROM:

Table

Description automatically generated

## DRIVES:

Table

Description automatically generated

## SUPPLIER:

Table

Description automatically generated

## REQUIRES:

Table

Description automatically generated

## NUTRIENT:

Table

Description automatically generated

## 16.UTILIZES:

Table

Description automatically generated with medium confidence

## 17. CUST\_PURCHASE\_DATES:

Table

Description automatically generated

## 18.EMP\_PHONE\_NUMBER:

Text

Description automatically generated with low confidence

## 19. SUPP\_SUPPLY\_DATES:

Table

Description automatically generated

# Queries:

## Query 1:

The nursery recently made a lot of profit and decided to award its hard-working employees. Increase the salary by 5% for all employees working in the ‘Agriculture’ department or supervising other employees.

SQL Query:

UPDATE EMPLOYEEE

SET SALARY=1.05\*Salary

WHERE ID IN (SELECT E.ID

FROM EMPLOYEE AS E, DEPARTMENT AS D

WHERE D.Name=’Agriculture’ AND E.DPT\_Nb=D.Department\_number)

UNION (SELECT E.ID

FROM EMPOYEE as E

WHERE E. Specialization=’Supervisor’);

OUTPUT:

Table

Description automatically generated

## Query 2:

A customer wanted to buy a plant from a branch in Beirut but had a specific budget. He cannot go over 100$ and is willing to pay at least 60$ for a plant. Retrieve all the information about plants which are stored in branches located in Beirut and whose prices ranges between 60 and 100 dollars.

SQL Query:

SELECT P.\*

FROM PLANT AS P, BRANCH AS B

WHERE B.City LIKE ‘Beirut%’ AND P.BRANCH\_Nb=B.Branch\_number AND (P.Price BETWEEN 60 AND 100);

OUTPUT:

Table

Description automatically generated

## Query 3:

Knowing how many plants need nutrient helps identify the amount of nutrients that the plant nursery must buy. Return the percentage of plants that require iron to grow.

SQL Query:

SELECT 100\*COUNT(\*)/(SELECT COUNT(\*) FROM PLANT) AS Percentage

FROM PLANT AS P, REQUIRES AS R, NUTRIENT AS N

WHERE R.Plant\_Code=P.Code AND R.Nutrient\_Code=N.Code AND N.Name=’Iron’

GROUP BY N.Name;

OUTPUT:



## Query 4:

The plant nursery needed urgently to contact the employees in ‘HR’ department. So, their supervisors were responsible for contacting them. However, some of them are not supervised by other employees. Retrieve the phone numbers and names of all employees working in ‘HR’ and are not supervised by other employees.

SQL Query:

SELECT E.F\_Name AS First Name, E.L\_Name AS Last Name, PN. Phone\_number AS Phone numbers

FROM EMPLOYEE AS E, EMP\_PHONE\_NUMBER AS PN, DEPARTMENT AS D

WHERE E.DPT\_Nb= D.Department\_number AND D.Name=’HR’ AND E.ID=PN. EMP\_ID AND E. Supervisor\_ID IS NULL;

OUTPUT:



## Query 5:

The equipment with serial numbers ‘KEG577302200145’ and ‘HRC770764478370’ were found to be stolen. So, the nursery decided to punish all employees who recently used both equipment. Show the expected decrease in salary of employees who used both the equipment if the decrease is equal to 2% of the sum of cost of both equipment.

SQL Query:

SELECT DISTINCT E.ID AS ID, E.Salary-0.02\*((SELECT Cost FROM EQUIPMENT WHERE S\_N= 'KEG577302200145')+( SELECT Cost FROM EQUIPMENT WHERE S\_N= 'HRC770764478370')) as New\_Salary

FROM EMPLOYEE E, EQUIPMENT

WHERE E.ID IN (SELECT UTILIZES.EMP\_ID

FROM UTILIZES

WHERE EQUIP\_S\_N= 'HRC770764478370'

INTERSECT

SELECT UTILIZES.EMP\_ID

FROM UTILIZES

WHERE EQUIP\_S\_N= 'KEG577302200145') AND (EQUIPMENT.S\_N= 'KEG577302200145' OR EQUIPMENT.S\_N= 'HRC770764478370');

OUTPUT:

Graphical user interface

Description automatically generated with medium confidence

## Query 6:

Young plants are destined to live longer than the old ones; thus, decreasing the price of old plants should be done in the branches so that customers would buy them before they die. Increase the price of plants ranging between 0 and 2 by 5%. Decrease the price of plants ranging between 3 and 4 by 5%. Decrease the price of plants ranging between 5 and 10 by 15%.

SQL Query:

UPDATE PLANT

SET Cost =

CASE WHEN Age BETWEEN 0 AND 2 THEN 1.05\*Cost

WHEN Age BETWEEN 3 AND 4 THEN 0.95\*Cost

WHEN Age BETWEEN 5 AND 7 THEN 0.85\*Cost;

OUTPUT:

Table

Description automatically generated

## Query 7:

The nursery decided to run some statistics regarding its sales in the month of September 2021. Retrieve the max, min and average number of plants purchased by a customer in September 2021.

SQL Query:

SELECT MIN(Quantity\_purchased), MAX(Quantity\_purchased), AVG(Quantity\_purchased)

FROM PURCHASES\_FROM P, CUST\_PURCHASE\_DATES D

WHERE P.CUST\_ID=D.CUST\_ID AND to\_char(D.Purchase\_dates, 'MM-YYYY')='09-2021';

OUTPUT:



## Query 8:

Due to the economic crisis, GOGREEN decided to kick some employees, so they decided to kick the old employees. To help GOGREEN with this decision, select the ID, and salary of all employees born in the 1970s and salary greater than 5000$.

SQL Query:

SELECT ID, Salary

FROM EMPLOYEE

WHERE ID IN (SELECT E.ID

FROM EMPLOYEE E

WHERE ID IN( SELECT E.ID FROM EMPLOYEE E WHERE TO\_CHAR(E.DOB, 'YYYY') LIKE '\_\_7\_') INTERSECT SELECT E.ID FROM EMPLOYEE E WHERE E.SALARY >5000);

OUTPUT:

Table

Description automatically generated

## Query 9:

Due to the fuel crisis, the nursery decided to help all employees working away from their homes. Retrieve the name of all employees who work in a department in Beirut but don’t live in Beirut and increase their salary by 5%.

SQL Query:

SELECT F\_NAME, L\_NAME, 1.05\*Salary as New\_Salary

FROM EMPLOYEE, DEPARTMENT

WHERE EMPLOYEE.City!='Beirut' AND DPT\_NB=DEPARTMENT\_NUMBER AND DEPARTMENT.City='Beirut';

OUTPUT:

Table

Description automatically generated

## Query 10:

There is a lack of human resources in the greenhouses. Therefore, the nursery plans on increasing the number of employees in it. To help them, retrieve the greenhouse numbers and name employees working there, if the greenhouse has less than 2 employees working there.

SQL Query:

SELECT E.F\_NAME AS FIRST\_NAME, E.L\_NAME AS LAST\_NAME, G.Greenhouse\_number AS GRN\_NBR

FROM EMPLOYEE E, GREENHOUSE G

WHERE EXISTS (SELECT COUNT(\*)

FROM EMPLOYEE

WHERE EMPLOYEE.GRN\_Nb=G.Greenhouse\_number

GROUP BY G.Greenhouse\_number

HAVING COUNT(\*)<2)AND E.GRN\_NB=G.Greenhouse\_number;

OUTPUT:

Graphical user interface, application

Description automatically generated

## Query 11:

SQL Query:

INSERT INTO CUSTOMER VALUES (202, NULL, 'Ramzi Harati', 'Hamra', NULL);

OUTPUT:



## O- Normalization Up to The BCNF Normal Form:

After creating all relations, we should improve them by normalizing according to several normal forms. Here we are going to normalize our database up to the fourth normal form which is the Boyce-Codd Normal Form. On each relation we are going to apply the four normal forms. We start with the first then second then third and at last the BCNF normal form. Let us first start by a general description to each normal form.

First Normal Form:

This form does not allow multivalued attributes, composite attributes, and their combinations to exist in a relation.

1. Only attribute values permitted are single atomic values.
2. Domain of an attribute must only include atomic values and the value of an attribute in a tuple must be a single value from the domain of that attribute.
3. Disallows having a set of values as an attribute value for a single tuple.

Second Normal Form:

The Second normal form is based on the concept of full functional dependency. Before explaining the second form let us define some concepts used in this form and other forms also.

Functional Dependencies: A constraint between two sets of attributes from the database. The values of the Y component of a tuple in relation R depend on, or are determined by the values of an X component. We say that Y is functionally dependent on X.

Prime attribute: An attribute that is a member of a candidate key in a relation R. An attribute is called non-prime if it is not a prime attribute that is, if it is not a member of any candidate key.

Full functional dependency: A functional dependency X →Y is a full functional dependency if removal of any attribute A from X means that the dependency does not hold anymore.

Partial Dependency: A functional dependency X →Y is a partial functional dependency if removal of any attribute A from X means that the dependency still holds.

A relation schema R is in the second normal form if every nonprime attribute in R is fully functionally dependent on every key of R and every nonprime attribute A in R is not partially dependent on any key in R.

Third Normal Form:

The third normal form is based on the concept of transitive dependency. So let us first define a transitive dependency.

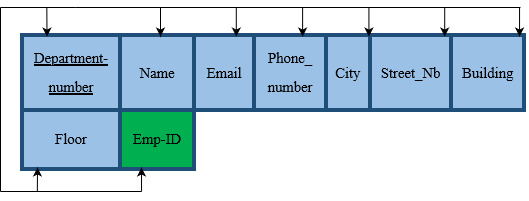
Transitive Dependency: A functional dependency X →Y in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X →Z and Z →Y hold.

A relation schema R is in the third normal form if it satisfies the second normal form, and no nonprime attribute of R is transitively dependent on the primary key. For every nontrivial functional dependency X →Y either X should be a super key or Y is a prime attribute.

Boycee-­Codd Normal Form:

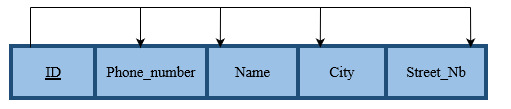
The Boycee-Codd normal form is a stricter form than the third normal form. The BCNF differs from the definition of the third normal form in only one condition. The third normal form allows the right hand side of the functional dependency to be a prime attribute while BCNF does not allow that.

## DEPARTMENT:



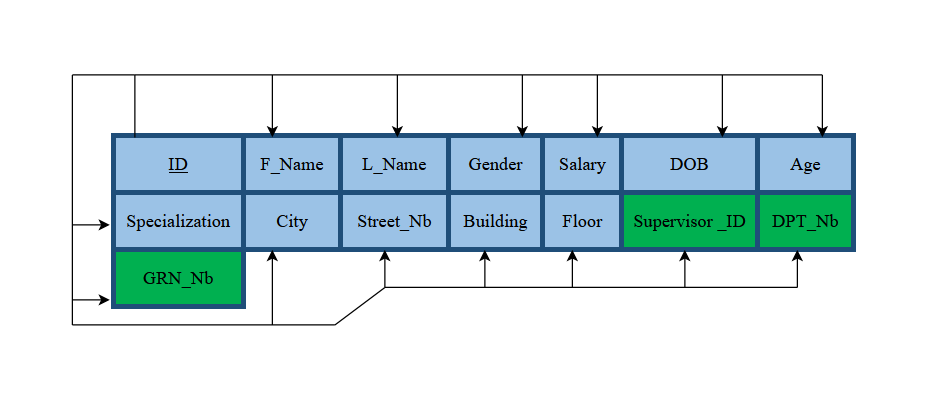
1. The **DEPARTMENT** relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.
2. The **DEPARTMENT** relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key **“Department-number”**.
3. The **DEPARTMENT** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“Department-number”**.
4. The **DEPARTMENT** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## CUSTOMER:



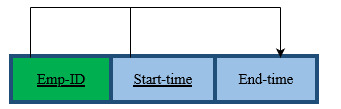
1. The **CUSTOMER** relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.
2. The **CUSTOMER** relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key **“ID”**.
3. The **CUSTOMER** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“ID”**.
4. The **CUSTOMER** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## EMPLOYEE:



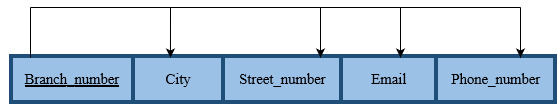
1. The **EMPLOYEE** relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.
2. The **EMPLOYEE** relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key **“ID”**.
3. The **EMPLOYEE** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“ID”**.
4. The **EMPLOYEE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## SCHEDULE:



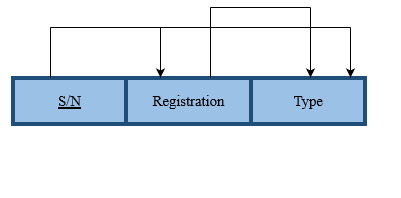
1. The **SCHEDULE** relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.
2. The **SCHEDULE** relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key **“{Emp-ID , Start-time}”**.
3. The **SCHEDULE** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“{Emp-ID , Start-time}”**.
4. The **SCHEDULE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## BRANCH:



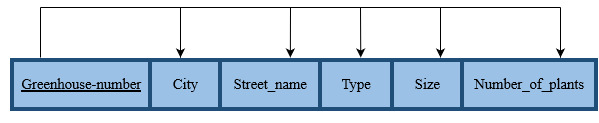
1. The **BRANCH** relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.
2. The **BRANCH** relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key **“Branch\_number”**.
3. The **BRANCH** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“Branch\_number”**.
4. The **BRANCH** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## VEHICLE:



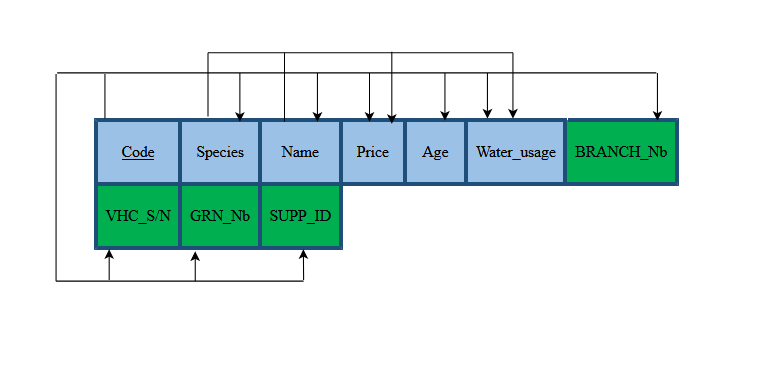
1. The **VEHICLE** relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.
2. The **VEHICLE** relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key **“S/N”**.
3. The **VEHICLE** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“S/N”**.
4. The **VEHICLE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## GREENHOUSE:



1. The **GREENHOUSE** relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.
2. The **GREENHOUSE** relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key **“Greenhouse-number”**.
3. The **GREENHOUSE** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“Greenhouse-number”**.
4. The **GREENHOUSE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## PLANT:

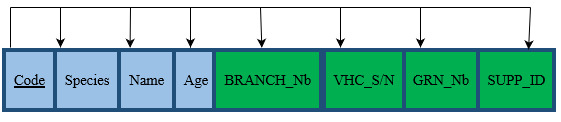


Our client would love it if we could get the price and Water\_usage of a plant using its species and name

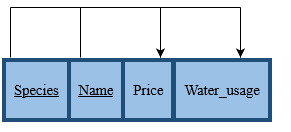
1. The **PLANT** relation schema satisfies all conditions of the 1NF because it has neither multi-valued attributes nor composite attributes. All attributes are single and atomic.
2. The **PLANT** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on “**Code**”.
3. Unfortunately, the 3NF doesn’t pass this test since {Species,Name} is not a superkey and in the same case the attributes its getting which are Price and Water\_usage prime attributes.

According to the client’s requirements we must construct two tables like below:

**PLANT1:**



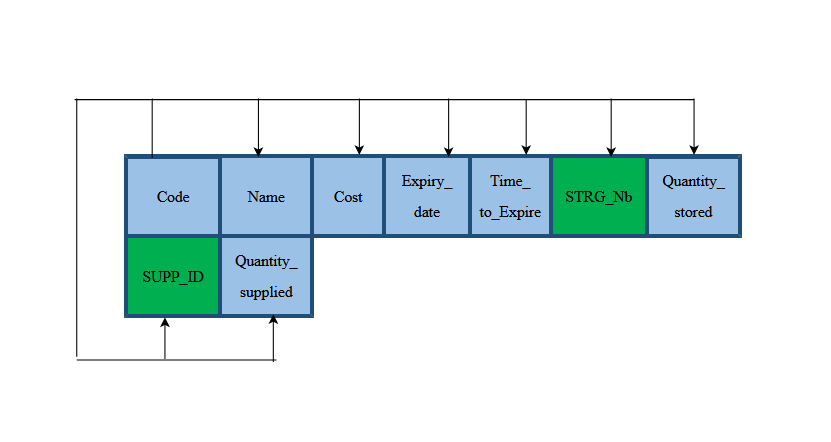
**PLANT2:**



According to table **PLANT2**, “Price” and “Water\_usage” are fully dependent on bothSpecies and Name thus satisfying the 3NF.

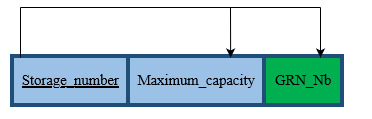
1. The **PLANT1** and **PLANT2** relation schemas satisfy all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## NUTRIENT:



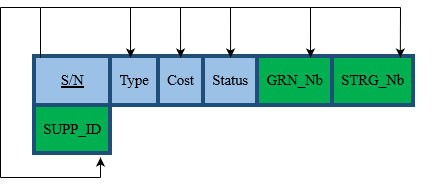
1. The **NUTRIENT** relation schema satisfies all conditions of the 1NF because it has neither multi-valued attributes nor composite attributes.
2. The **NUTRIENT** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on “Code”.
3. The **NUTRIENT** relation schema satisfies all conditions of the 3NF since it satisfies 2NF and since there are no non-prime attributes that are transitively dependent on “Code”.
4. The **NUTREINT** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## STORAGE:



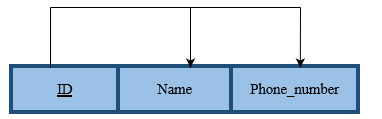
1. The **STORAGE** relation schema satisfies all conditions of the 1NF because it has neither multi-valued attributes nor composite attributes.
2. The **STORAGE** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on “Storage\_number”.
3. The **STORAGE** relation schema satisfies all conditions of the 3NF since it satisfies 2NF and since there are no non-prime attributes that are transitively dependent on “Storage\_number”.
4. The **STORAGE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## EQUIPMENT:



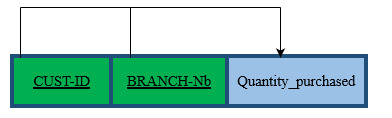
1. The **EQUIPMENT** relation schema satisfies all conditions of the 1NF because it has neither multi-valued attributes nor composite attributes.
2. The **EQUIPMENT** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on “S/N”.
3. The **EQUIPMENT** relation schema satisfies all conditions of the 3NF since it satisfies 2NF and since there are no non-prime attributes that are transitively dependent on “S/N”.
4. The **EQUIPMENT** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## SUPPLIER:



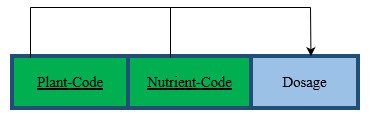
1. The **SUPPLIER** relation schema satisfies all conditions of the 1NF because it has neither multi-valued attributes nor composite attributes.
2. The **SUPPLIER** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on “ID”.
3. The **SUPPLIER** relation schema satisfies all conditions of the 3NF since it satisfies 2NF and since there are no non-prime attributes that are transitively dependent on “ID”.
4. The **SUPPLIER** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## PURCHASES\_FROM:



1. The **PURCHASES\_FROM** relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.
2. The **PURCHASES\_FROM** relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key **“{CUST-ID, BRANCH-Nb}”**.
3. The **PURCHASES\_FROM** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“{CUST-ID, BRANCH-Nb}”**.
4. The **PURCHASES\_FROM** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

## REQUIRES:



1. The **REQUIRES** relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.
2. The **REQUIRES** relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key **“{Plant\_Code, Nutrient-Code}”**.
3. The **REQUIRES** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“{Plant\_Code, Nutrient-Code}”**.
4. The **REQUIRES** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

Relation Schemas without non-prime attributes:

## DRIVES:

|  |  |
| --- | --- |
| EMP-ID | VHC-S/N |

## UTILIZES:

|  |  |
| --- | --- |
| EMP-ID | EQUIP-S/N |

## CUST\_PURCHASE\_DATES:

|  |  |
| --- | --- |
| CUST-ID | Purchase-dates |

## EMP\_PHONE\_NUMBER:

|  |  |
| --- | --- |
| EMP-ID | Phone-  number |

## SUPP\_SUPPLY\_DATES:

|  |  |
| --- | --- |
| SUPP-ID | Supply-dates |

# P- Conclusion:

A database is crucial to the establishment and functioning of an organization, containing hundreds of data from staff members, departments, and clients. In our case, this plant nursery database is used to show how plants are grown in greenhouses and taken care of by the employees specialized in the field. Plants are being sold in multiple retail branches that are related to the plant nursery itself. Moreover, this database classifies each component needed to do certain tasks, such as delivering plants to the retail branches via vehicles. It also shows the number of nutrients that each plant needs (such as nutrients and water), and this displays where equipment and nutrients are being stored. Additionally, this database takes into consideration the client's point of view via a customer-focused entity type that will get to know the clients better. Lastly, this database provides schedules for each employee, which will prevent time conflicts and time management errors.

# Q- Instructor’s Comments and Evaluation:

This page should be filled by our instructor DR. RAMZI HARATY regarding any comments, and improvements for our database system. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_