***Ara Department of Computing***



Assignment Cover Sheet

|  |  |
| --- | --- |
| **Student Name:** | ZILIN LI |
| **Student ID:** | 99172142 |
|  |  |
| **Course Code & Year / Semester:** | BCDE301 |
| **Assignment:** | Assessment #1 - Database Design for |
|  | Wholey Moley Foods |
|  |  |
| **Tutor(s):** |  |

|  |  |
| --- | --- |
| **Word Count:** |  |
| **Date:** |  |

I \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (student/s signature/s) hereby declare that this assignment is all my/our own work, and any sources are referenced and cited.

BCDE103 Database Design

Assignment One:

Database Design for Wholey Moley Foods

Student: Zilin Li

# 1. Introduction

The client requested for designing a data management system for his business. The data management system is required to track merchandise sales, employee information and customer information.

This report shows the database structure, database entity selection, key selection, data type selection, entity connectivity/relationships, database attributes, extent of normalization, etc.

# 2. Entities

## **2.1** Entities **selection**

The entities selection of the database in this project is based on the following three factors:

* User requirements and specifications: Entities such as ‘Customer’, ‘Account’, ‘Branch’, ‘CustomerOrder’, ‘Product’, and ‘Employee’ are identified based on user requirements and specifications.
* Analysis of entity attributes: Based on the analysis of entity attributes, five new entities composed of original multi-attribute components were established, such as ‘CustomerType’, ‘PaymentType’, ‘SalePerMonth’, ‘Skill’, and ‘Allergen’.
* Relationship between the entities: Based on the relationship between the entities, five bridge entities and one additional entity are established.

The following figure will describe the entities in detail:

|  |  |  |
| --- | --- | --- |
| No. | Entities | Description |
| 1 | Customer | Used to store customer information. |
| 2 | Account | Used to store customer account information. |
| 3 | Branch | Used to store branch information. |
| 4 | CustomerOrder | Used to store order information. |
| 5 | Product | Used to store product information. |
| 6 | Employee | Used to store employee information. |
| 7 | CustomerType | Used to store customer types. |
| 8 | PaymentType | Used to store payment types. |
| 9 | SalePerMonth | Used to store total sales amount per month. |
| 10 | Skill | Used to store employee skills. |
| 11 | Allergen | Used to store product allergens. |
| 12 | OrderDetail | Used to connect customer orders and products. |
| 13 | BranchProduct | Used to connect branches and products. |
| 14 | ProductAllergen | Used to connect allergens and products. |
| 15 | EmployeeSkill | Used to connect employees and skills. |
| 16 | WorkBranch | Used to connect employees and branches. |
| 17 | ProductCombo | Used to handle unary relationships that exist in the product entity |

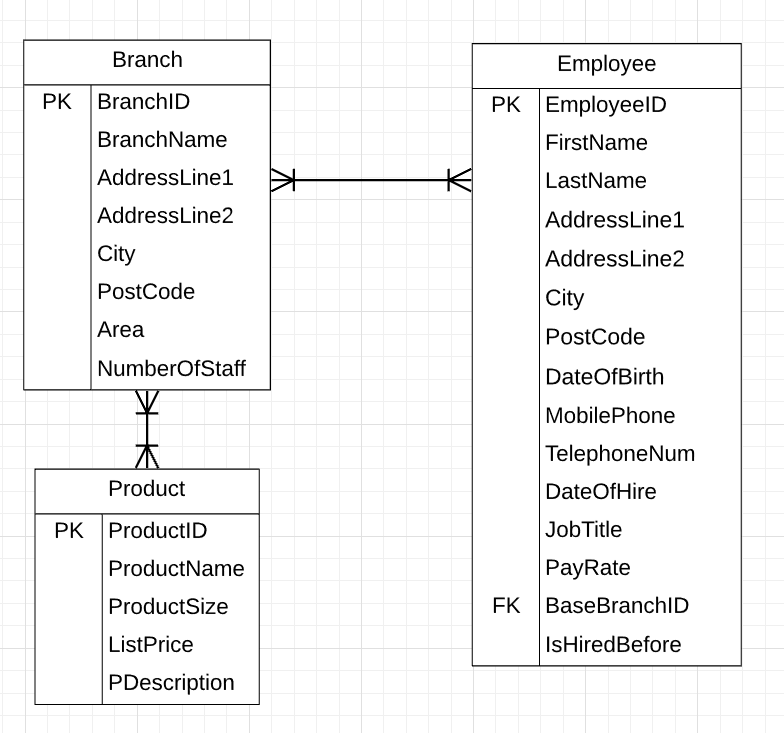
## **2.2 Bridging entities**

Bridging entities, also known as composite entities, are primarily used to establish connections between other entities in order to eliminate the effects of many-to-many relationships between entities.

For example, Client wants to track employee information and product information in each branch. To do this, we need entities with branches, employees, and products.

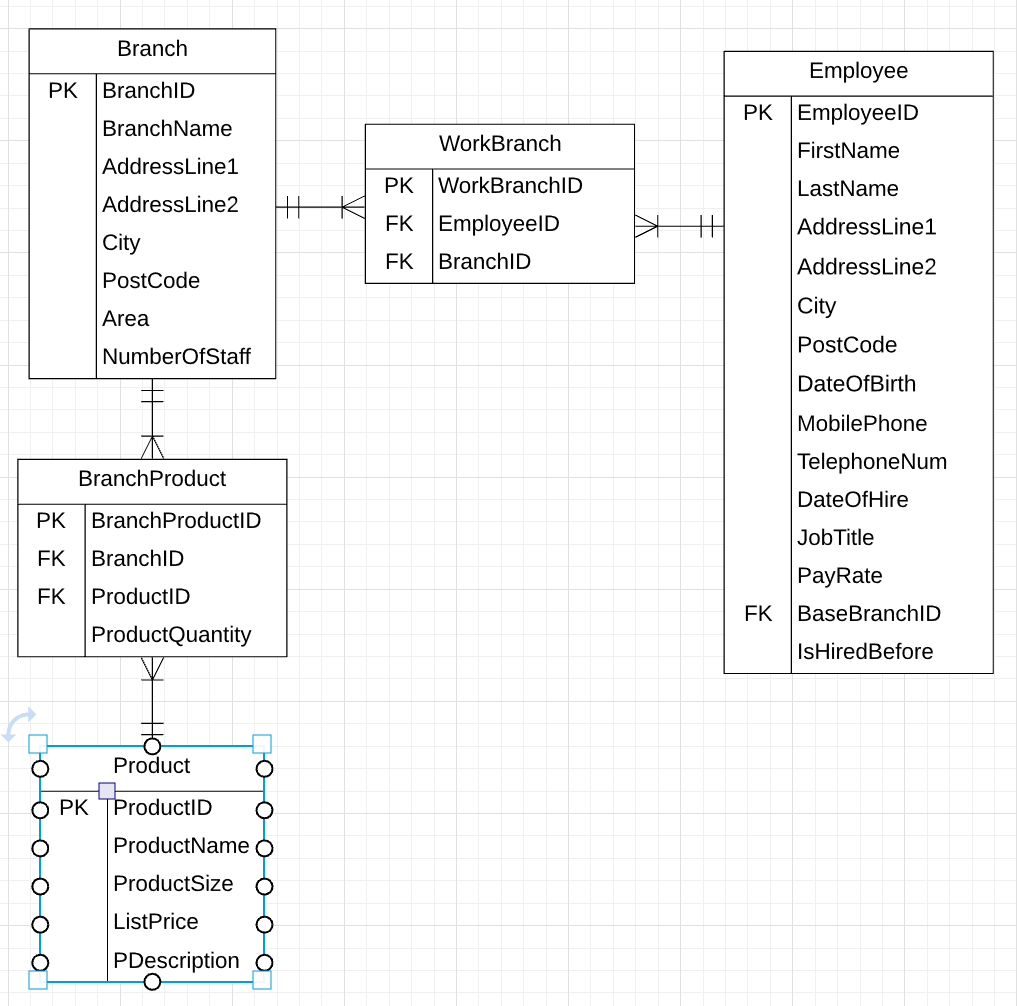
The relationship is as follows:

* A branch can have many employees, and an employee can work in different branches.
* A branch can have many different products, and the same products can exist in different branches.



When we draw the ER diagram, we get two of many-to-many relationships. These relationships are indeed correct. We can know which employees work in the branch and which products exist in the branch through the branch structure, but we cannot know the detailed information of a specific employee or product through the branch structure, and vice versa.

Therefore, we introduced bridging entities into the above many-to-many relations, and solved the problem of one-to-one correspondence of information.



need a table example.

In this database, entities ‘OrderDetail’, ‘BranchProduct’, ‘ProductAllergen’, ‘EmployeeSkill’ and ‘WorkBranch’ are all bridge entities.

## 2.3 supertype/subtype

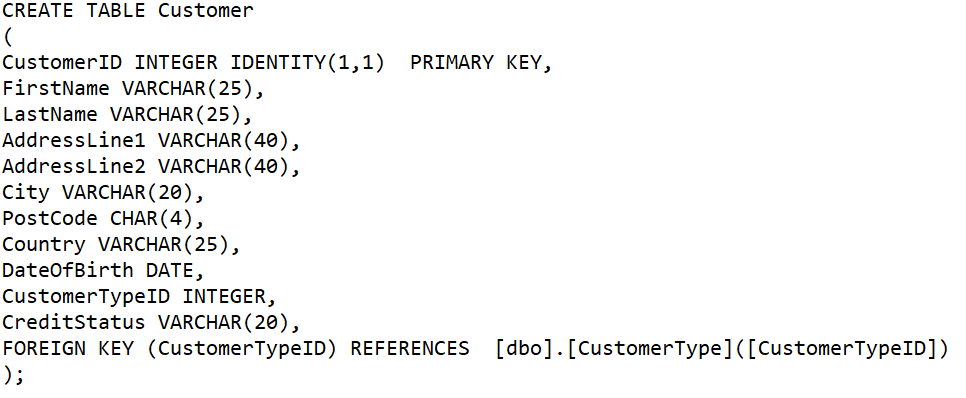
## 2.4 weak entity

# 3. Keys

## 3.1 Primary key

The primary key is used to identify each row in the table, and the value of the primary key is unique in the table. All tables in this database have unique primary keys. Since there is no special user requirement for primary key types, to maintain the uniqueness of primary keys, we use self-increment integer for all primary key types.

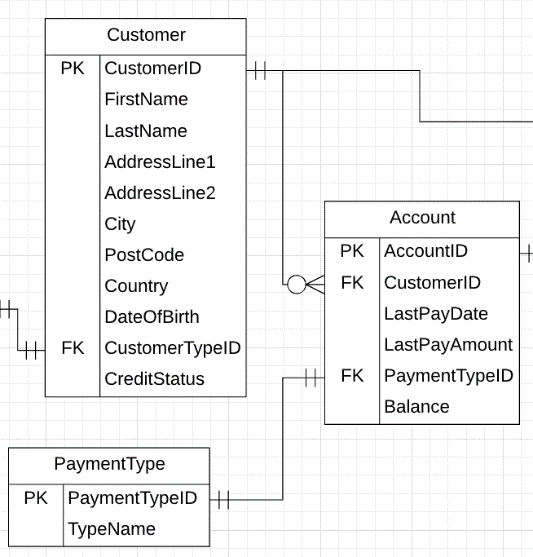
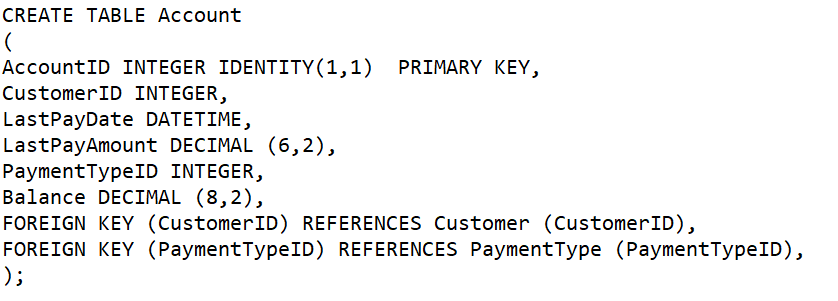
For example: Customer table’s primary key is CustomerID.

## 3.2 Foreign key

A foreign key is used to create an association between two tables, usually the primary key of another table. In the design of this database, we use a lot of foreign keys to help establish the connection of database tables.

For example: In Account table, there are two foreign keys. Foreign key, ‘CustomerID’, is used to establish the connection between Account table and Customer table, and it is also the primary key of Customer table. Foreign key, ‘PaymentTypeID’, is used to establish the connection between Account table and PaymentType table, and it is also the primary key of PaymentType table.

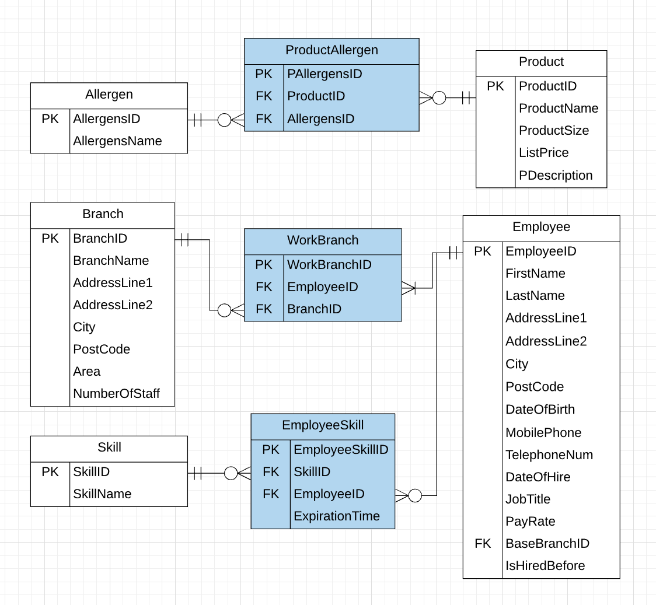
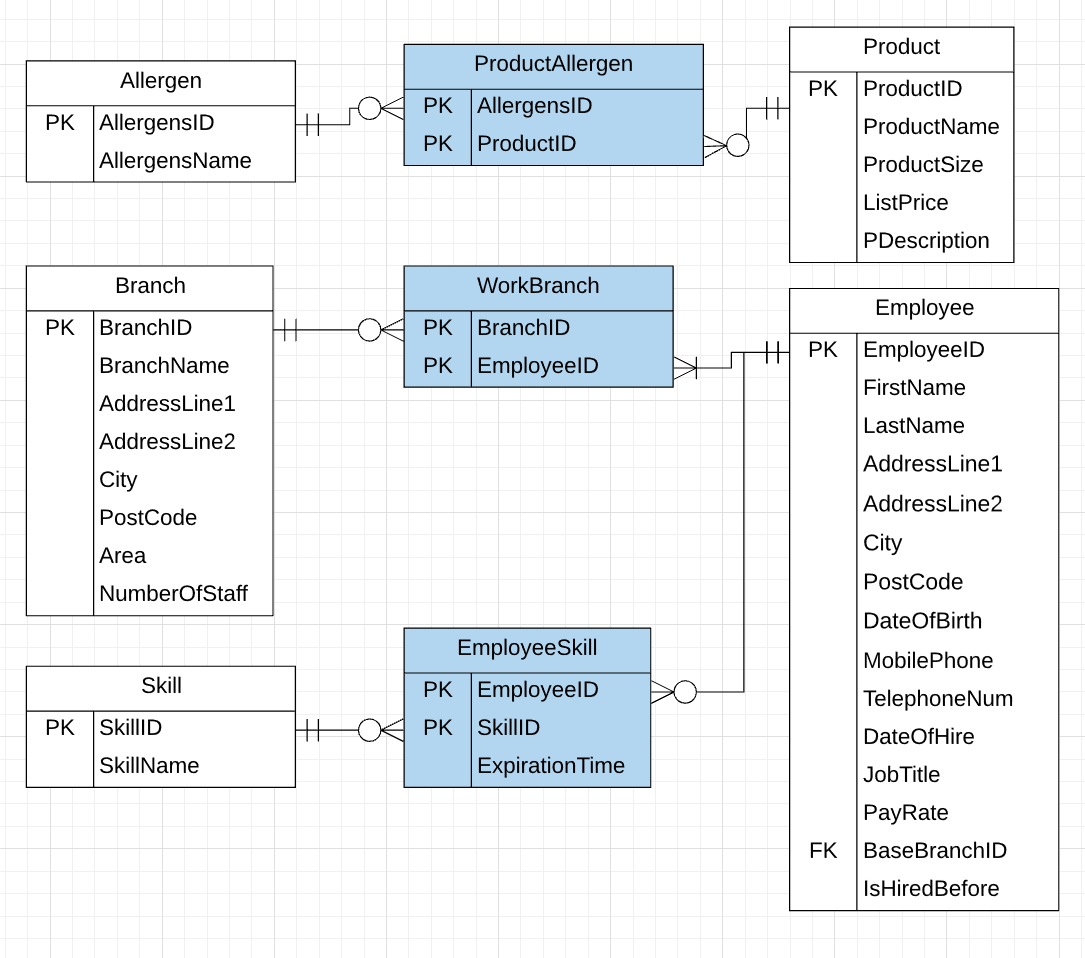


## 3.3 Composite key

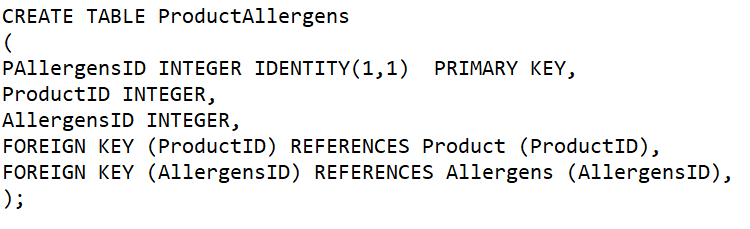
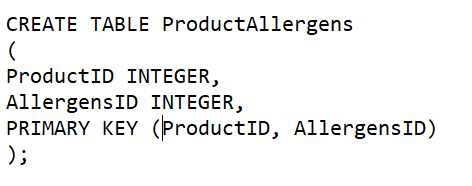
Composite key is a combination of two or more columns that uniquely identify rows in a table. We did not use composite keys in the design of this database, but we can still discuss the possibility of using composite keys in the database.

For example: In this database, there are some composite entities in which we can choose to use composite keys.

The original plan: The optional plan:

Coding example:

Alternative：

# 4. Attributes

All the attributes involved in this database are added according to user requirements.

## 4.1 Derived attribute

A derived attribute is an attribute whose value is calculated (derived) from other attributes. The derived attribute need not be physically stored within the database; instead, it can be derived by using an algorithm. For example, **employee age**, **number of staff members** and **total monthly sales of the branch**.

When we design the attribute, we use the attribute of the date of birth instead of the age. In addition, considering the complexity of the query and the importance of the data, we specially established the ‘SalePerMonth’ as an entity to facilitate our client to query and track the monthly sales records, and established the ‘NumberOfStaff’ attributes for easier query.

Coding example:

## 4.2 Composite attribute

A composite attribute is an attribute that can be further subdivided to yield additional attributes.

For example, the attribute ‘Address’ can be subdivided into street, city, state, and zip code. The attribute ‘Name’ can be subdivided into first name and last name.

In this database design, in addition to '**address**' and '**name**', **'phone number'** also belongs to the composite property. Considering that the customer base and employees are all local users, the composition of phone Numbers is relatively uniform, so the ‘phone number’ are not further separated.

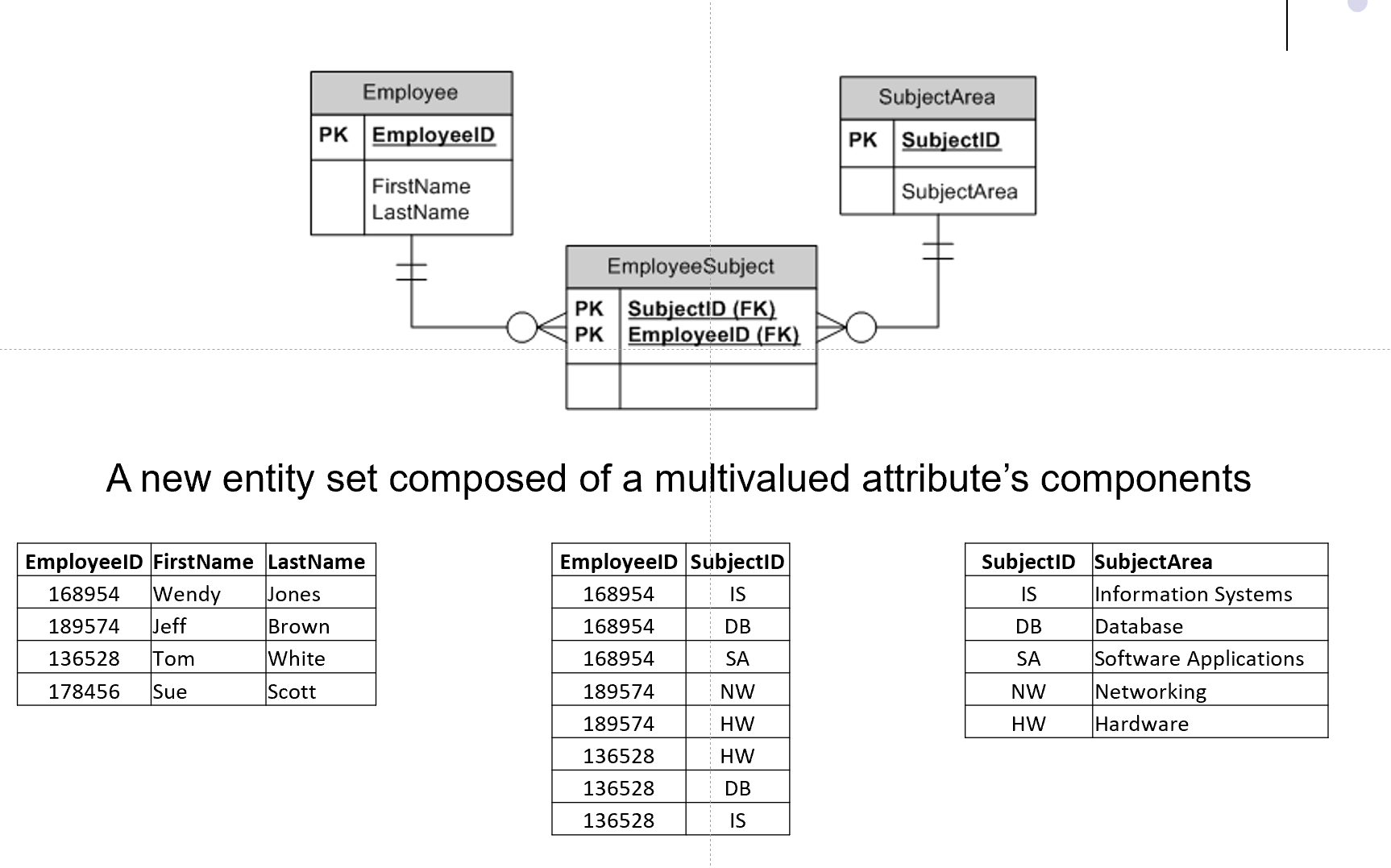
Coding example:

## 4.3 Multivalued attributes

Multi valued attributes are attributes that can have many values.

For example: ‘**Employee**’ attribute in the Branch table, **‘Skill’** attribute in the Employee table, **‘Allergen’** attribute in the Product table, etc.

In the design of this database, we mainly solve this problem by creating a new entity composed of the original multivalued attribute’s components.

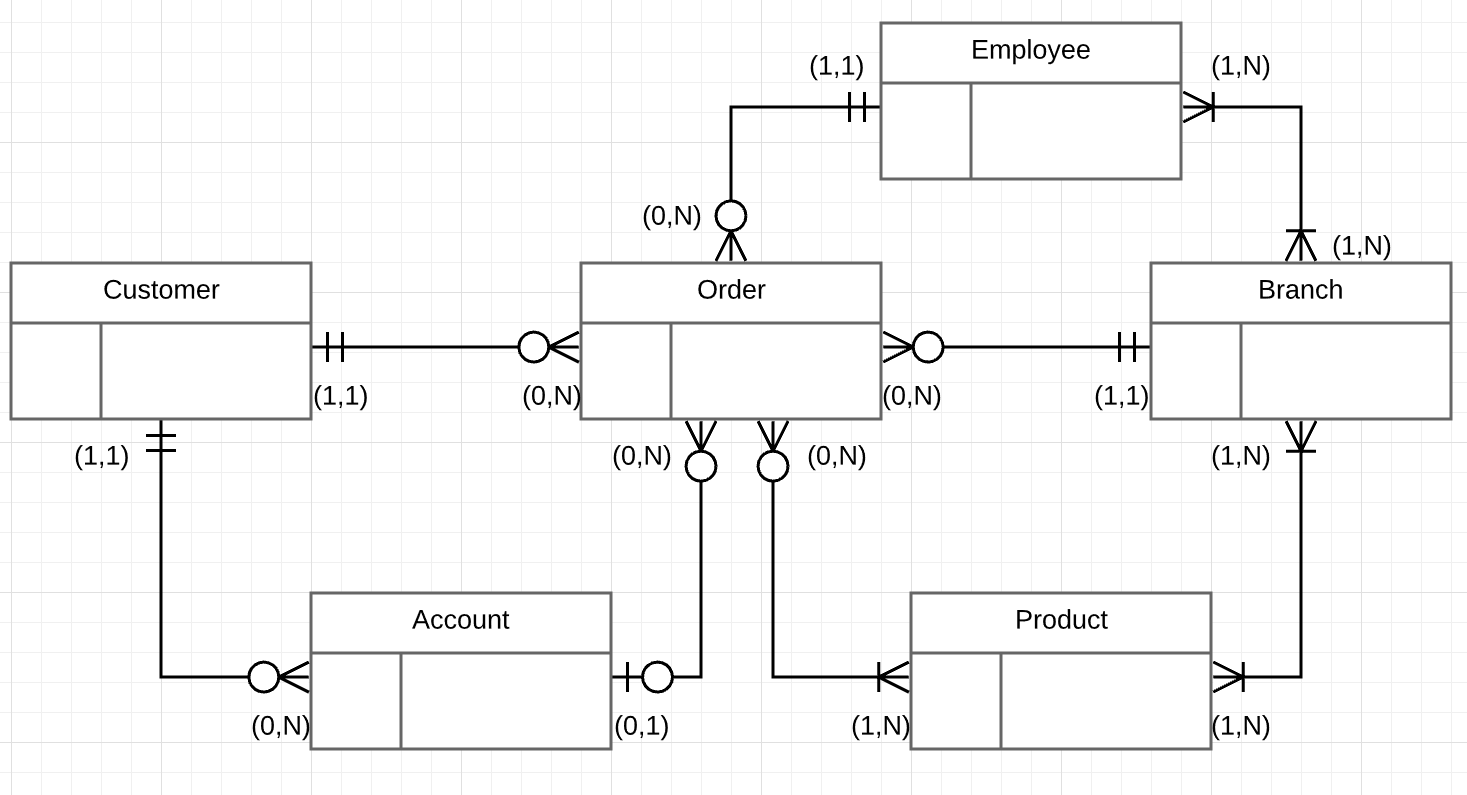


# 5. Connectivity/Relationships

Relationships are used to describe the relationships between entities. In this project, we first analyzed what are the main entities in the database, and then analyzed the relationships among the entities through entity relationship diagrams to further improve the design of the database.

## 5.1 ERD

## 5.2 The relationship between the main entities



* Customer & Order: A customer may have many orders or no order. An order has one and only one user.
* Customer & Account: A customer may have many accounts or no account. An account has one and only one customer.
* Account & Order: An account can submit multiple orders or no orders, and an order can be submitted through an account or not through an account.
* Order & Employee: An order has one and only one sales employee, and an employee can have multiple orders or no orders.
* Order & Branch: Orders can only be submitted in one and only one branch, and one branch can involve multiple orders or no orders.
* Order & Product: An order may contain at least one item or more items. An item may exist in more than one order or may not exist in any order.
* Product & Branch: Products are stored in at least one or more branches. A branch has at least one or more products.
* Branch & Employee: An employee works in at least one or more branches. A branch has at least one or more employees.

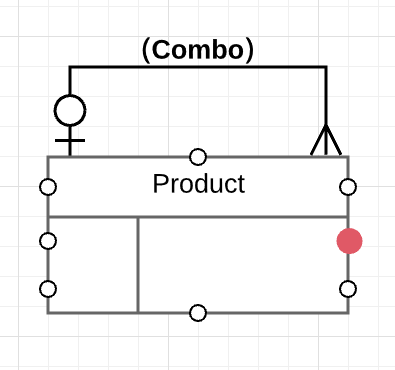
## **Relationship Degree**

Degree of relationship is the number of entities sets that are participated in that relationship. Based on the degree, the relationships may be identified as unary, binary, ternary and so on.

Below we will discuss the relationship degree that occurs in this project.

**Unary relationship:**

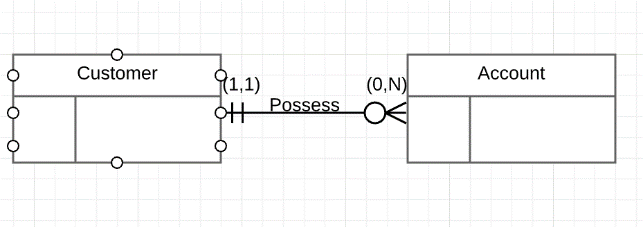
According to the client’s requirement, a product can be composed of multiple products.



* A product may belong to a product portfolio that contains multiple products.
* A product may belong to more than one product portfolio.
* A product may not be part of any product portfolio.

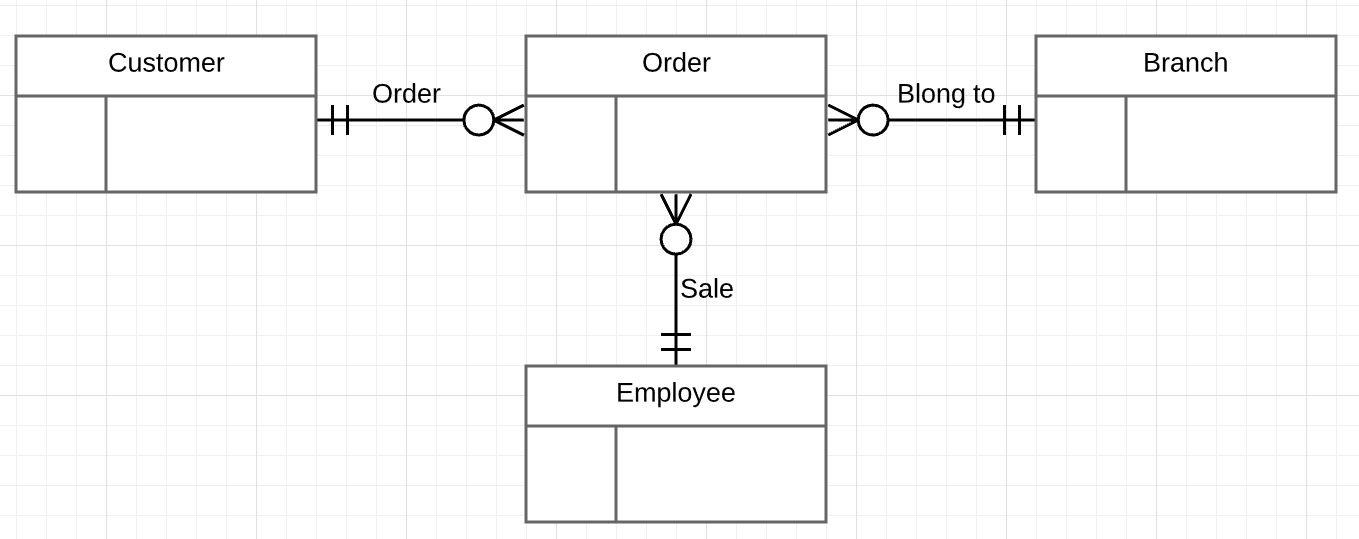
**Binary relationship:**

In the following relationship, only Customer entity and Account entity are associated



**Ternary relationship:**

In the following relationship, there are three entities are associated.



# 6.Extent of normalization