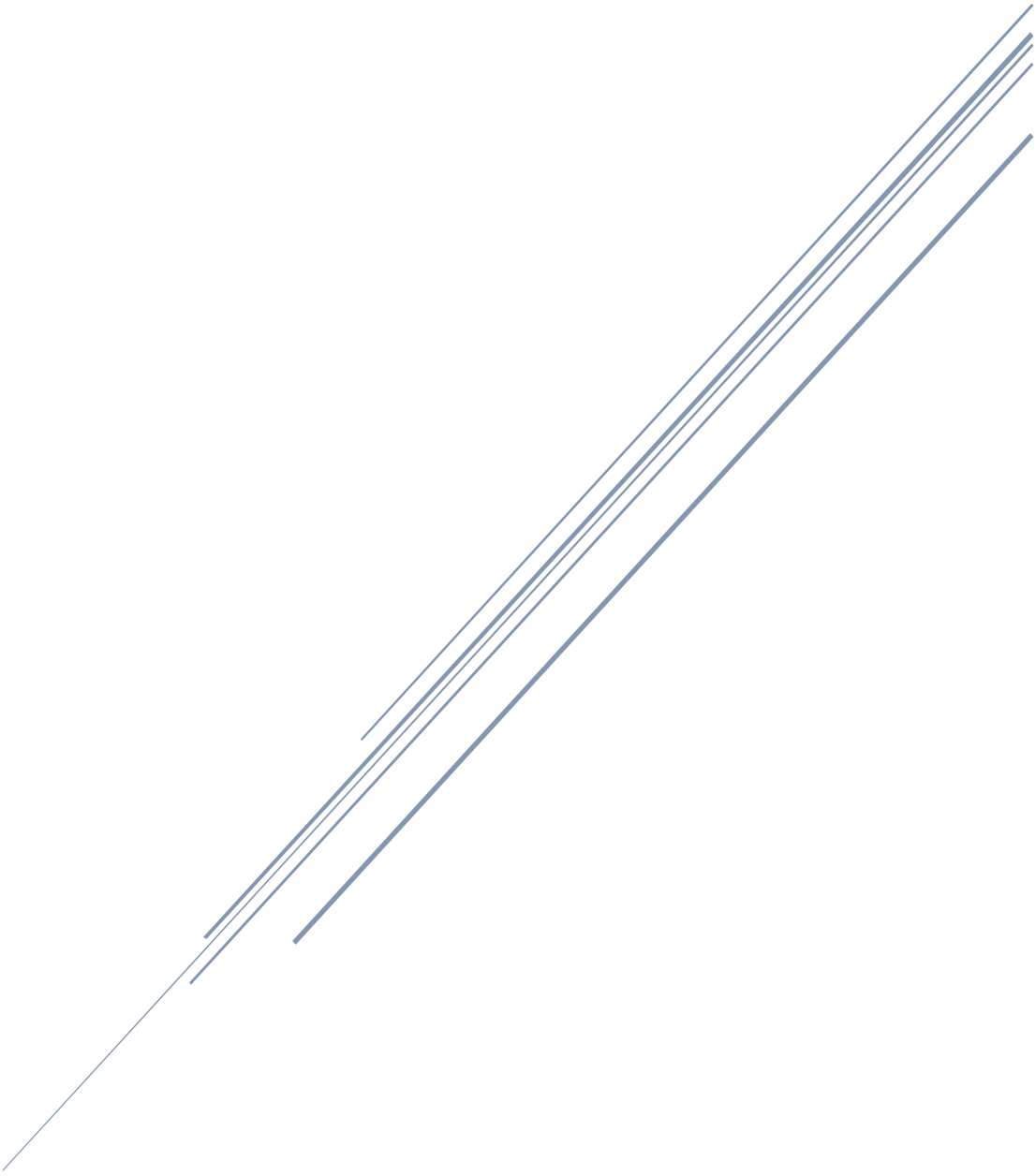
DATABASE DESIGN PROCESS AND ISSUES REPORT

For Wholey Moley Foods



Stella Fu

04 September 2020

Table of Contents

1. Introduction

This report is developed to record the process of designing a database management system for Wholey Moley Foods. The business focuses on healthy food. The products are sold via three branches, farmers markets and trade shows. Alister Lance is the owner of the business. He would like to use the designed system to manage the sales of finished items, customers and employees.

Microsoft Visio is utilised to design the Entity Relationship Diagram (ERD). SQL Server Management Studio is the tool for ERD implementation and data testing.

The following sections of the report will introduce the design process and issues regarding entities, keys, bridging entities, connectivity, attributes and normalisation.

2. Design Process and Issues

This section shows how the designer created a relational schema from the aspects of entities, primary and foreign keys, composite entities, relationships, attributes and extent of normalisation.

2.1 Entities

According to the provided business rules, six basic entities were generated in the first stage. They are **Customer**, **Employee**, **Branch**, **Item**, **Account and Order**.

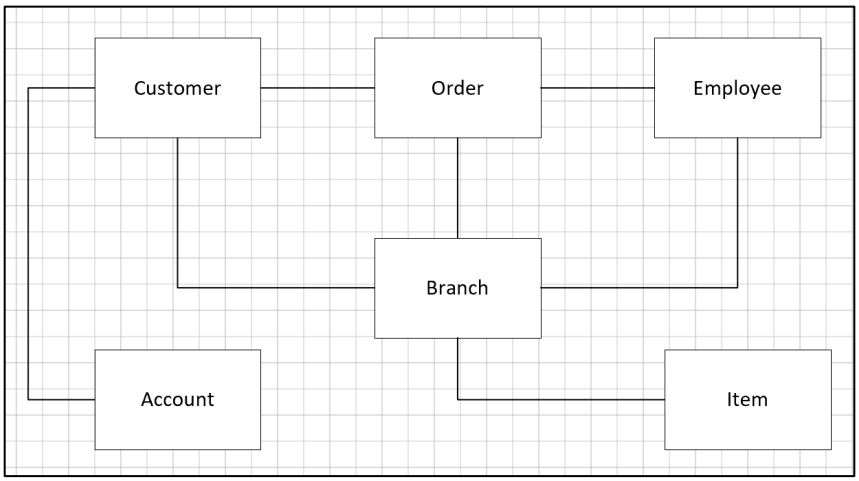


Figure 1 - Six Basic Entities at the First Stage

2.2 Keys

At this point of time, each of the six entities has a primary key that is unique. They are **cutomerID**, **orderID**, **employID**, **branchID**, **itemID**, and **accountID**.

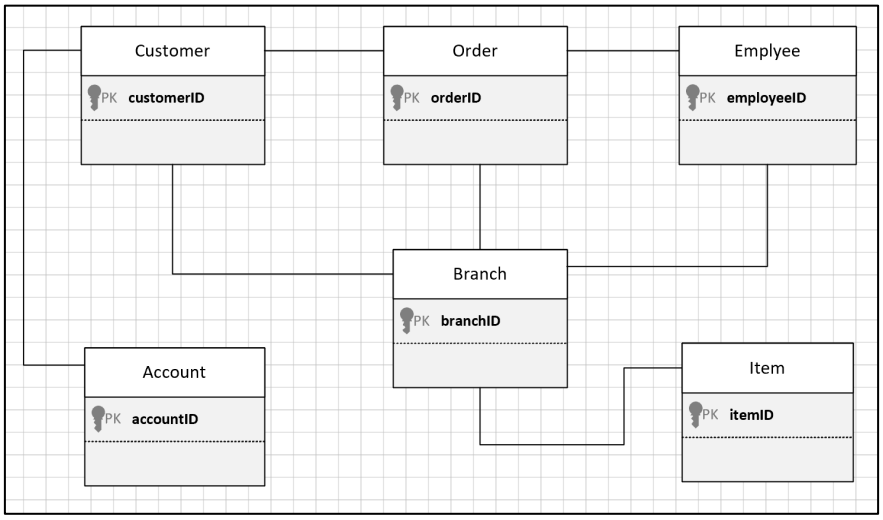


Figure 2 - Basic Entities with Primary Key

2.3 Relationships

Relationships between the six basic entities are showed with **cardinality notation (Crow's Foot Notation)** in the diagram below. This type of cardinality notation was applied throughout the whole ERD design.

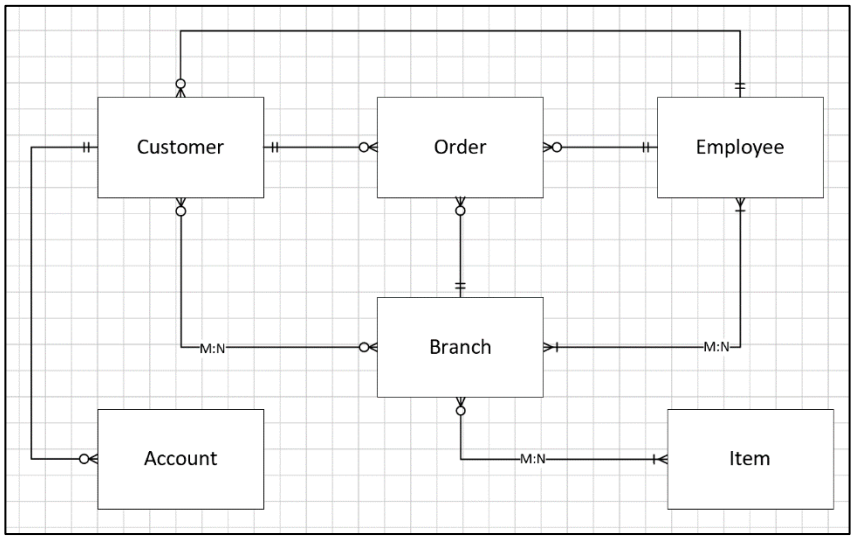


Figure 3 - Six Basic Entities

- A customer may have zero, one or several accounts but an account always matches only one customer
- A customer is served by one employee and an employee can have zero or many customers
- A customer may place many orders, but an order belongs to one and only one customer
- An order has one and only one employee who receives the order, and an employee may have many orders
- An employee is based in one branch and he/she may work at multiple branches, and a branch can have many employees
- A branch can have many orders, but an order belongs to one and only branch
- A branch may have many customers and a customer may shop at many branches
- A branch has one or many items and an item may be unavailable in some branches or available in many branches

The cardinality notation used in the above diagram is explained in the following table for reference.

	One and Only One (1:1)
	One or Many (1:M)
	Zero or Many (0:M)

Figure 4 - Cardinality Notation

2.4 Bridging Entities

Three many-to-many (M:N) relationships were encountered in the *Figure 3* diagram.

1. A customer may visit many branches over the years and a branch can have many customers
2. A branch may sell many items or certain items, and an item is available in many branches or a certain branch
3. An employee is based in a branch, but he/she may work at multiple branches, and a branch can have many employees

However, the relational model cannot model many-to-many (M:N) relationships. To solve the issues, bridges must be created between the entities that display M:N relationship.

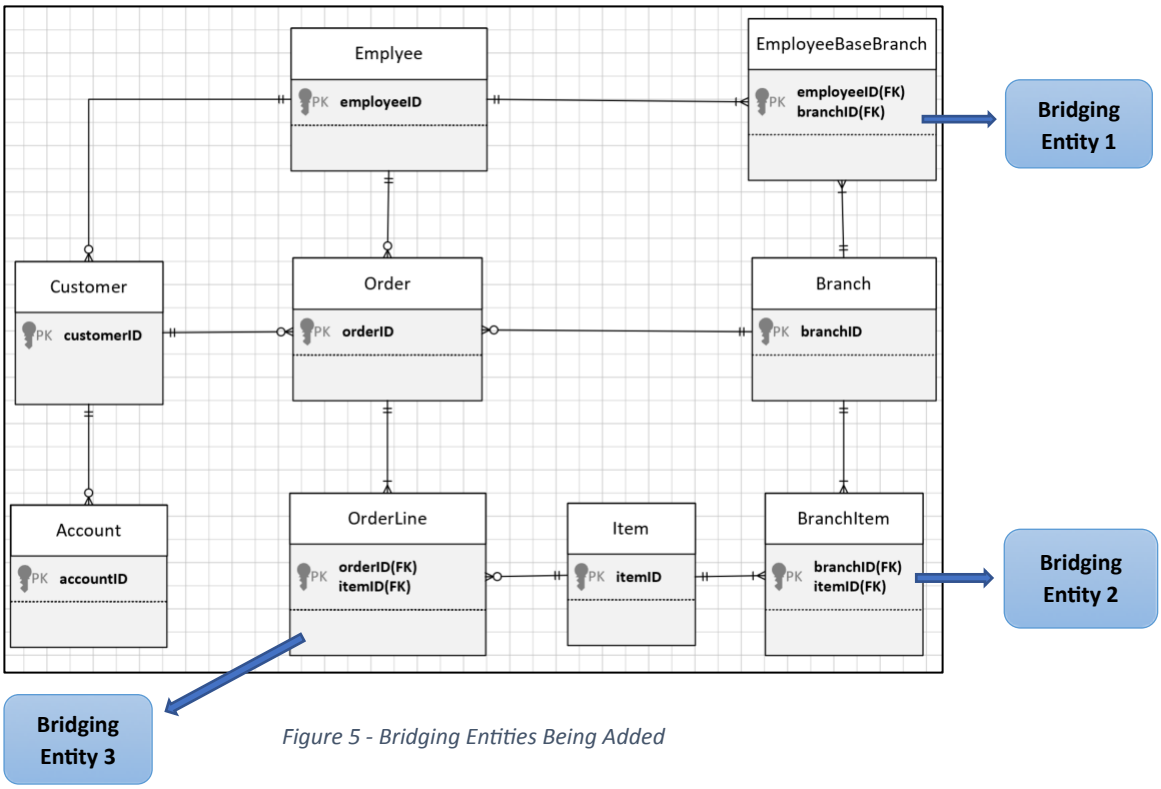


Figure 5 - Bridging Entities Being Added

In each of the bridging entities in *Figure 5*, the unique primary key consists of two primary keys from each of the entities to be connected. In another word, the bridging entities used foreign keys as their primary key. At this point of time, we have nine entities and there are none many-to-many relationships between those entities (see *Figure 5*). However, in the following section, more bridging entities or composite entities will be created.

2.5 Attributes

Based on the nine established entities, attributes were added to every entity according to the early provided business rules and the later add-on requirements from the Forum on Moodle.

Specifically, a single-valued attribute can have only one single value. A simple attribute cannot be subdivided. A composite attribute can be further subdivided into additional attributes. Therefore, some new attributes were created to subdivide the composite attributes.

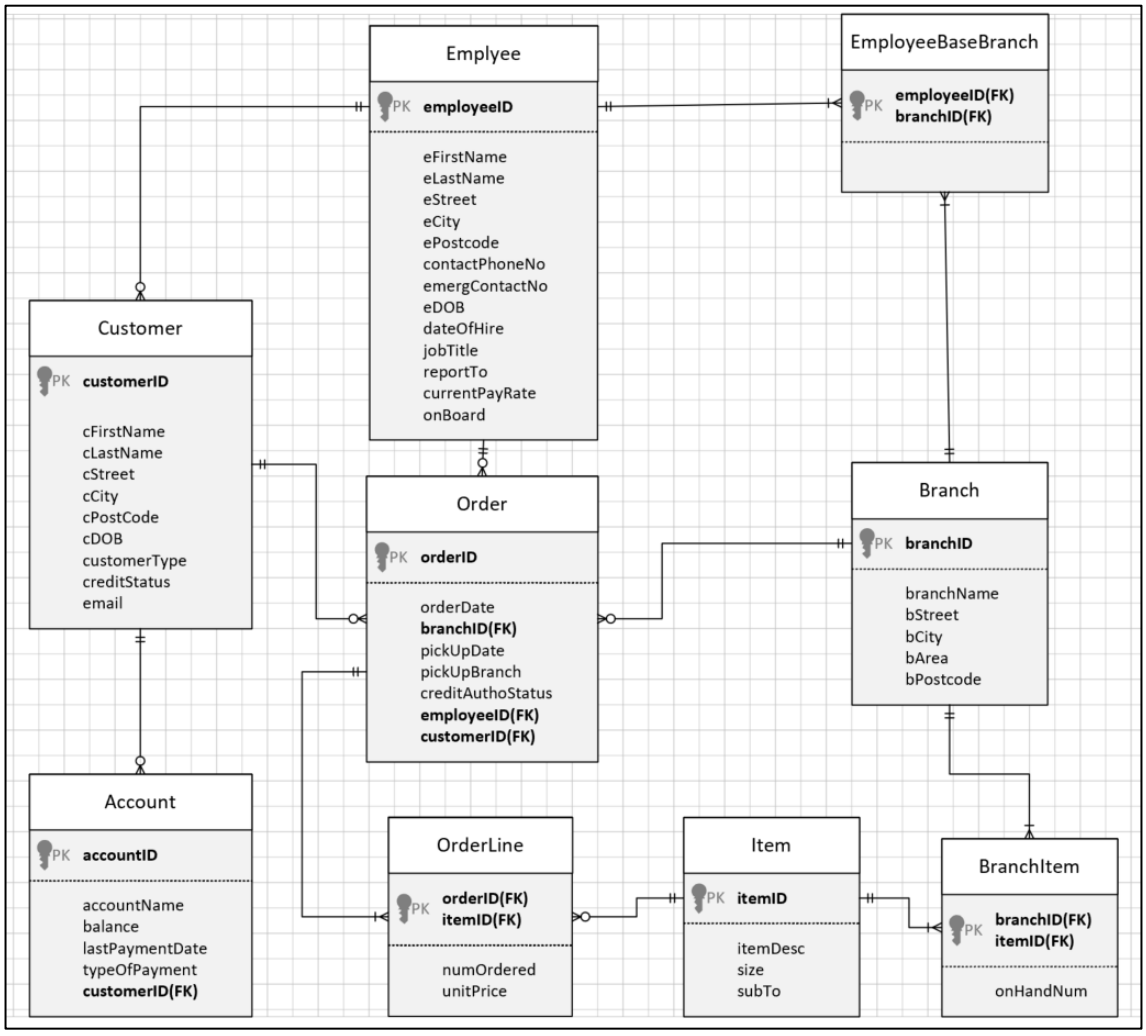


Figure 6 - Nine Entities with Attributes

2.5.1 Customer Table

Simple attribute(i.e.)	Composite attribute(i.e.)	Single-valued attribute(i.e.)
Date of birth(DOB)	Name	Customer ID
Customer type	Address	
Credit status		
Email		

Figure 7 - Attributes in Customer Table

Composite attributes in the above table can be subdivided into attributes:

- Name – First Name, Last Name
- Address – Street, City, Postcode

2.5.2 Employee Table

Simple attribute(i.e.)	Composite attribute(i.e.)	Single-valued attribute(i.e.)	Multivalued attribute(i.e.)
Date of birth(DOB)	Name	Employee ID	*Telephone numbers
Date of hire	Address		*Skill sets
Job title			*Working hours
Current pay rate			
*ReportTo			
*OnBoard			

Figure 8 - Attributes in Employee Table

Employees' age can be derived from the data of their date of birth and the current date, so 'Age' is a derived attribute and it will not be stored in the database.

Composite attributes in the above table can be subdivided into attributes:

- Name – First Name, Last Name
- Address – Street, City, Postcode

According to the client's requirement of keeping records of:

- Employees that have previously been used (onboard status)
- History of employees' working hours, with starting and finishing time for the purpose of staff lunch break and contact tracing (working hour)
- Skill sets of employees (skill set)
- Both employee phone numbers and emergency contact phone numbers (telephone numbers)

Specifically, in the Employee table, stating 'Yes' or 'No' in the onBoard column instead of deleting the employee from the table if they are leaving would be an easy-to-keep record of employees that have been used.

A new table named WorkingHour needs to be created because one employee can have many working hours in a branch or branches. These are repeatable information that do not need to be recorded under the Employee table.

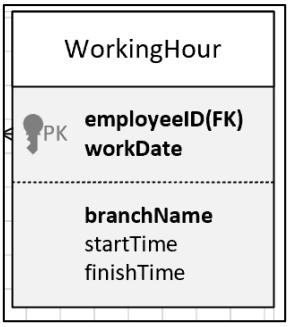


Figure 9 - WorkingHour Table

However, according to the client's reply to employee working hours on Moodle, the designer decided to record the branch name instead of the branch ID in this table, because their staff normally talk about the branch name rather than a branch ID. If the owner would like to have a report to see who worked at a certain branch on a certain date, we can make an SQL query by using the '%' before and after a required branch name.

Moreover, the client is concerned about the repeatable information of skill sets. Therefore, a new table named Skillset is created to fulfil this requirement. However, the relationship between the Employee table and Skillset table is many-to-many (M:N), so a bridge entity called EmployeeSkill was created to solve this issue. The primary key is composed of two foreign keys. They are the primary keys from the connected tables.

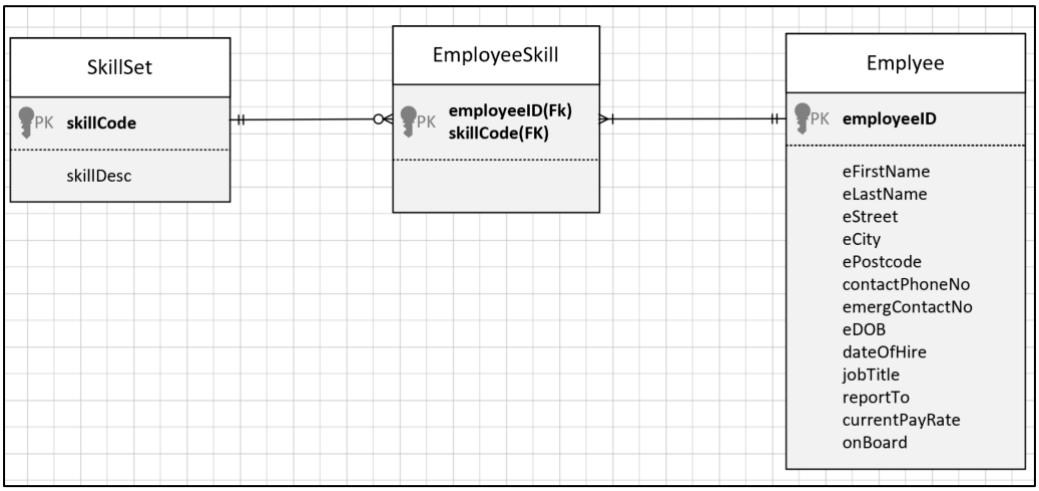


Figure 10 - EmployeeSkill Entity

Further, an attribute named reportTo was added to the Employee table to enable the client to produce a report that displays the employee's name and their manager's name in future operations. This need can be achieved by making SQL queries of recursion.

Lastly, according to the client's further requirement for an employee's phone number, the multivalued attribute: telephone numbers was divided into two attributes as below.

- Employee phone number
- Emergency contact phone number

2.5.3 Branch Table

Simple	Composite	Single-valued	Multivalued
Branch name	Address	Branch ID	*Monthly total sales amount
Area			

Figure 11 - Attributes in Branch Table

The composite attribute: Address can be subdivided into attributes:

- Street, City and Postcode

According to the information provided by the client, they do have branch IDs that consist of letters and numbers. For example, WASH1 is the branch ID for Washdyke Farmer's Market. Therefore, the letter + number became the format of Branch ID, and the testing data will follow this naming rule for Branch ID.

Also, the business is operated in the following areas at this point of time:

- Canterbury, Hanan Shields, Timaru, Oamaru

To avoid redundant information in the table, the monthly total sales amount will not become an attribute. As it can be gained via making SQL query by using relevant attributes from other tables.

2.5.4 Item Table

Simple attribute(i.e.)	Single-valued attribute(i.e.)	Multivalued attribute(i.e.)
Item description	Item ID	Allergen
Size		
*SubTo		

Figure 12 - Attributes in Item Table

An item may have one or more allergens, so another table called Allergen was created. However, the relationship between the Item table and the Allergen table is many-to-many (M:N), so a bridge entity called ItemAllergen was created to solve this problem. Two foreign keys from the connected tables became the unique primary key in the ItemAllergen Table.

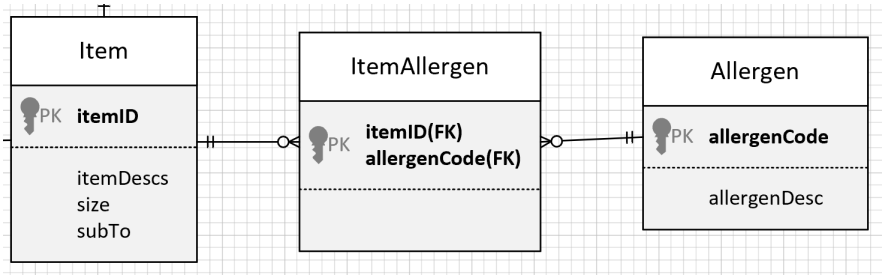


Figure 13 - ItemAllergen Entity

Particularly, an item can be composed of several items. For example, item 2001 may include item 256 and item 259. In this case, item 256 and item 259 all sub to item 2001. Avoid creating another table to record repeatable information, an attribute name subTo was added to the Item Table.

2.5.5 Order Table

Simple attribute(i.e.)	**Single-valued attribute(i.e.)**
Order date	Order ID
PickUp date	Branch ID
Credit authorisation status	Employee ID

Figure 14 - Attributes in Order Table

However, an order can have one or many items and an item can be sold through many orders. This is a many-to-many (M:N) relationship, so a composite entity named OrderLine was created to solve this issue (refer to Figure 15).

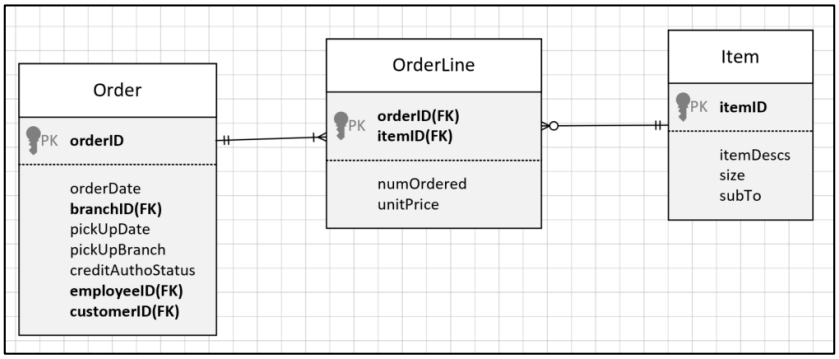


Figure 15 - OrderLine Entity

2.6 Extent of Normalisation

This section shows one example of normalisation for this design work. Primary keys are those blocks with a blue background.

2.6.1 First Normal Form

In the first normal form, repeatable groups should be removed, primary keys need to be defined and attributes need to be single-valued. Therefore, 'name' was subdivided into first name and last name. 'address' was subdivided into street, city and postcode.

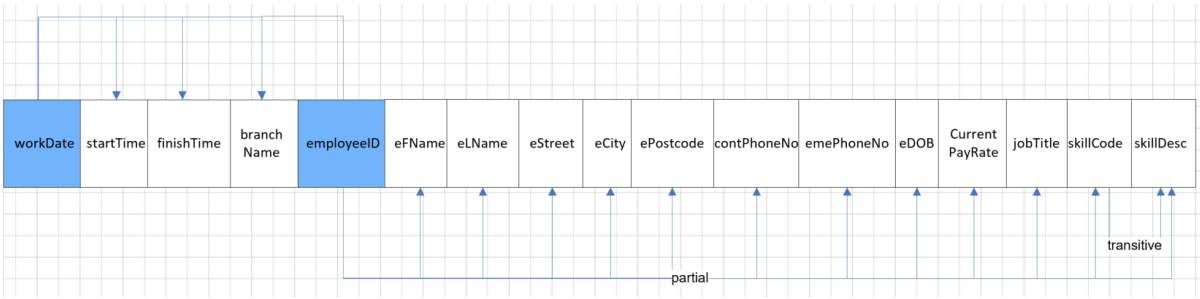


Figure 16 - An Example of the First Normal Form

Partial dependencies that rely on part of the primary key are marked in *Figure 16*.

Also, skill code determines skill description, but this is a transitive dependency because the skill code attribute is a non-primary key.

2.6.2 Second Normal Form

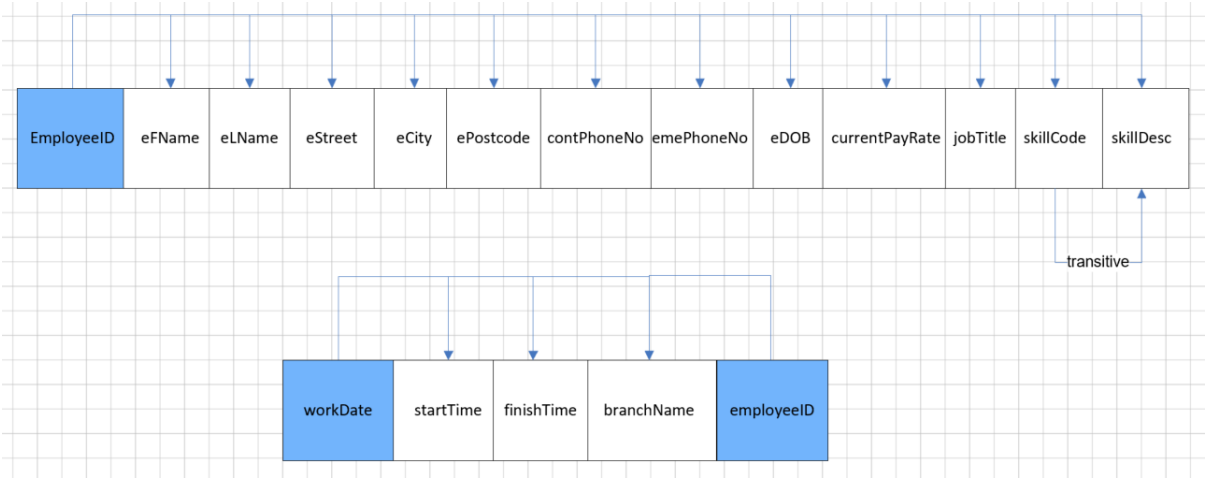


Figure 17 - An Example of the Second Normal Form

In the second normal form, the table should be in the first normal form, and the partial dependencies need to be removed.

2.6.3 Third Normal Form

Figure 18 - An Example of the Third Normal Form

In the third normal form, the table is in the second normal form and excludes transitive dependency.

2.6.4 ERD

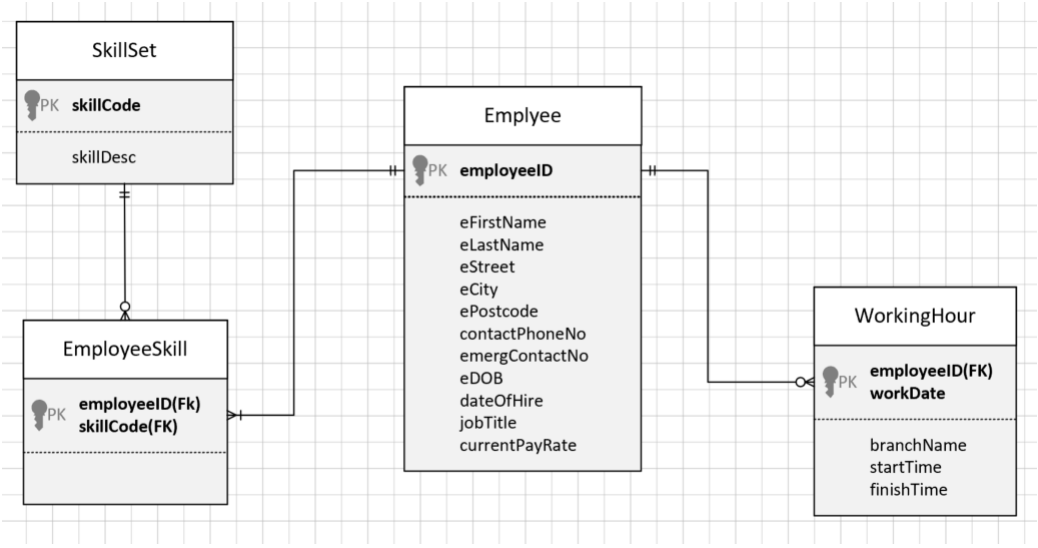


Figure 19 - An Example of ERD

After applying the third normal form, an ERD was obtained and displayed in *Figure 19*.

2.7 The Extended ERD

The following relational schema was created by using Microsoft Visio.

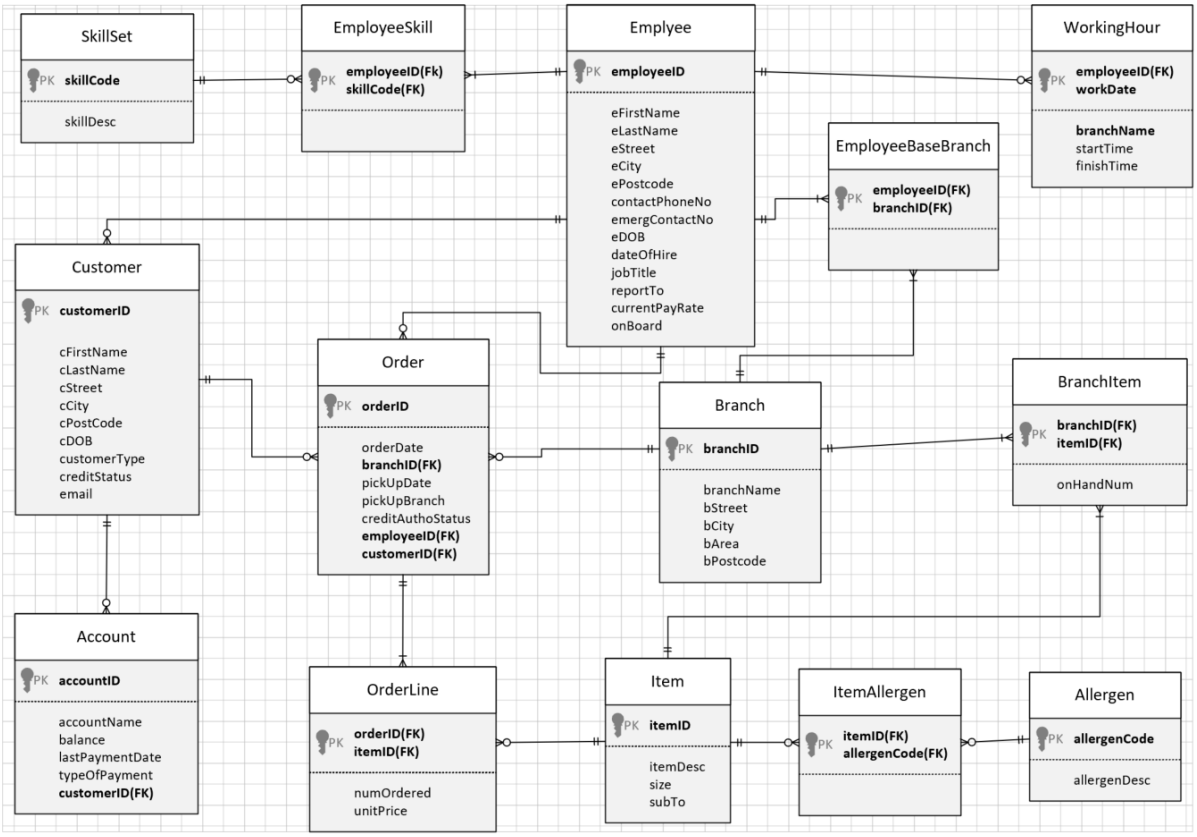


Figure 20 – The Extended ERD with Entities, Relationships, Cardinality, Attributes and Primary Keys

2.8 Data Testing in SQL Server Management Studio

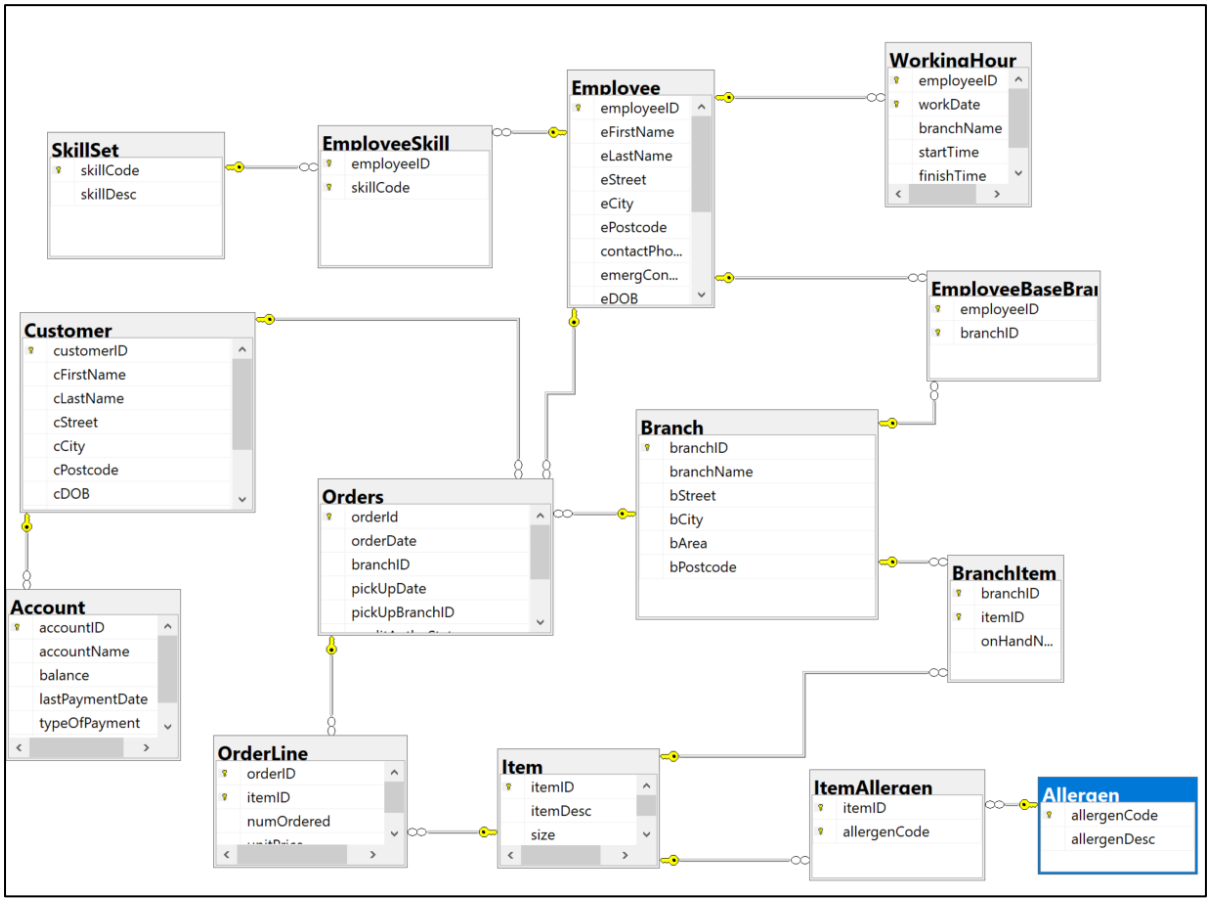


Figure 21 - Database Diagram in SQL Server

3. Conclusion and Recommendation

In summary, this report recorded the process and the issues encountered as well as how to solve those issues during designing the database management system for the business of Wholey Moley Foods.

Eventually, 14 tables were created with relevant attributes to achieve the client's requirements. But at this stage, with the limited scope and time, the designed database only includes the business's selling part. The manufacturing and purchasing parts are out of the scope of this design at the moment.

By using the Microsoft Visio and SQL Server Management Studio, the extended ERD and database diagram was created and attached to this report.