ECM2419: Database Theory & Design Coursework

Contents

1 Database Design 2

1.1 Entity-Relationship Diagram 2

1.2 Explanation 2

1.2.1 The ‘Branch’ and ‘Office’ Tables 2

1.2.2 The ‘Employee’, ‘Branch Manager’, and ‘Salesperson’ Tables 2

1.2.3 The ‘Customer’ Table 3

1.2.4 The ‘Customer Employee’ Table 3

1.2.5 The ‘Manufacturer’ and ‘Product’ Tables 3

1.2.6 The ‘Sells’ Relationship 4

2 Normalisation 4

2.1 Table Definitions 4

2.1.1 Relational Schema 4

2.1.2 Relational Instances 4

2.2 Explanation 5

Declaration

AI-supported use is permitted in this assessment. I acknowledge the following uses of GenAI tools in this assessment:

- I have not used any GenAI tools in preparing this assessment.

I declare that I have referenced use of GenAI outputs within my assessment in line with the University referencing guidelines.

1 Database Design

1.1 Entity-Relationship Diagram

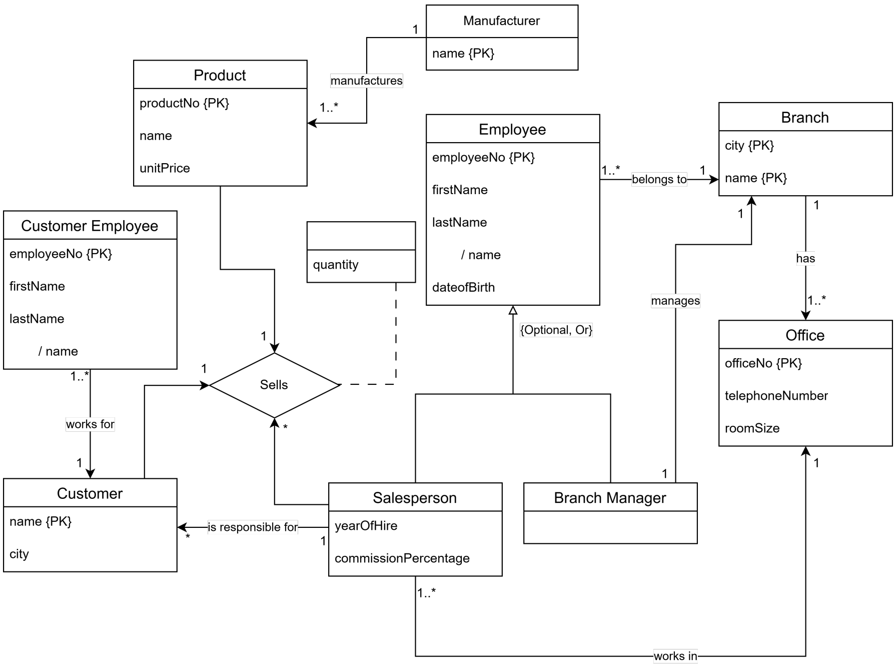

Figure 1: Entity-Relationship Diagram for C&E Hardware.

1.2 Explanation

1.2.1 The 'Branch' and 'Office' Tables

In addition to their city, I also gave `Branch` a `name` field to differentiate between branches in the same city. This is because it is possible that a city may have multiple branches, but they would be named differently. The `city` and `name` fields then form a composite primary key for the `Branch` table.

`officeNo` was an obvious choice for the primary key of the `Office` table, as it is unique to each office. The `Office` table additionally has the required attributes for `telephoneNo` and `roomSize`.

The `Branch` table has a 'has' relationship with the `Office` table, with a cardinality of `1:1..*` because a branch can have many, but at least one, offices, while an office can only belong to one branch.

1.2.2 The 'Employee', 'Branch Manager', and 'Salesperson' Tables

I chose to give the `Employee` a `employeeNo` for their primary key, as it cannot be guaranteed that all employees in a branch will have unique names. Although only a 'salesperson number' is mentioned in the specification, I have assumed that managers and other employees also have unique numbers which can be collectively referred to as `employeeNo`.

The Branch Manager and Salesperson tables are specialisations of the Employee table, allowing them to share attributes such as employeeNo, firstName, lastName, and dateOfBirth. This specialisation is {Optional, Or} because an employee must be either a branch manager, salesperson, or neither.

An Employee has a 'works in' relationship with a Branch. This relationship has a cardinality of 1:1..* because it is assumed that an employee works in exactly one branch, while a branch can have many (but at least one) employees.

A Branch Manager 'manages' exactly one Branch, and a Branch has exactly one Branch Manager. The Branch Manager does not have a relation to a specific Office, as they are assumed to manage the entire branch rather than a specific office.

1.2.3 The 'Customer' Table

I chose to use name as the primary key for the Customer table, as it should be unique to each Customer. The table also has the required attribute of city. The Customer table has a 'has' relationship with the Customer Employee table, with a cardinality of 1:1..* because a customer must have at least one employee, but an employee can work for many customers.

The Salesperson table has a 'is responsible for' relationship with a Customer with a cardinality of 1:* because a salesperson can be responsible for many (including zero) customers, but a customer must have exactly one salesperson.

1.2.4 The 'Customer Employee' Table

Instead of reusing the Employee table for employees of customers as well as C&E Hardware, I chose to create a separate Customer Employee table. This is because the data about a customer's employee is only relevant as long as the customer remains a customer of C&E Hardware, and so by separating the tables, deleting a Customer can cascade to delete all of its employees. This also allows a Customer Employee to have less personal data than a Customer, as such information is not relevant to C&E Hardware.

A customerNo is assigned to each Customer Employee as their primary key, as it is unique to each employee. The employee's firstName and lastName are also stored in this table.

The Customer Employee table has a 'works for' relationship with the Customer table, with a cardinality of 1:1..* because an employee must work for a customer, but a customer can have many employees.

1.2.5 The 'Manufacturer' and 'Product' Tables

The Product table has a productNo as its primary key, as it is unique to each product. The table also has the required attributes of name and unitPrice.

The Manufacturer table has a name as its primary key, as it is unique to each manufacturer.

The Product table has a 'manufactures' relationship with the Manufacturer table, with a cardinality of 1:1..* because a product must be manufactured by a manufacturer, but a manufacturer can manufacture many products.

1.2.6 The 'Sells' Relationship

There is a three-way relationship between a `Salesperson`, `Customer`, and `Product`. This relationship additionally has an attribute of `quantity`. I have assumed that a branch or office does not have a specific list of products, but rather that a salesperson can sell any product from any manufacturer. This is why the relationship is between the salesperson and the product, rather than the branch and the product.

The cardinality of the `Sells` relation is chosen because each relationship must contain exactly one salesperson, one customer, and one product, but a salesperson can sell many products to many customers.

2 Normalisation

2.1 Table Definitions

2.1.1 Relational Schema

The normalised relational schema is as follows:

```
Salesperson(SalespersonNumber, SalespersonName, CommissionPercentage,
    ↪ YearOfHire, DepartmentNumber)
Primary Key: SalespersonNumber
Foreign Key: DepartmentNumber references Department(DepartmentNumber)

Department(DepartmentNumber, ManagerName)
Primary Key: DepartmentNumber

Product(ProductNumber, ProductName, UnitPrice)
Primary Key: ProductNumber

ProductSale(SalespersonNumber, ProductNumber, Quantity)
Primary Key: SalespersonNumber, ProductNumber
Foreign Key: SalespersonNumber references Salesperson(SalespersonNumber),
    ↪ ProductNumber references Product(ProductNumber)
```

2.1.2 Relational Instances

The relational instances are given in tables 1, 2, 3, and 4; where the primary key is in underlined and foreign keys are in italics[1].

SalespersonNo	**SalespersonName**	**CommissionPcnt**	**YrOfHire**	***DeptNo***
37	Baker	10	2015	73
86	Adams	15	2011	59
14	Johnson	10	2018	59
61	Davies	20	2011	59

Table 1: The `Salesperson` instance.

[1] Some attribute names have been shortened from those given in the schema.

DeptNo	**ManagerName**
73	Scott
59	Lopez

Table 2: The `Department` instance.

ProductNo	**ProductName**	**UnitPrice**
6722	Pliers	11.50
4013	Saw	26.25
9440	Hammer	17.50
6386	Wrench	12.95
1765	Drill	32.99

Table 3: The `Product` instance.

ProductNo	***SalespersonNo***	**Quantity**
6722	37	688
4013	37	170
9440	37	473
6386	86	1745
9440	86	2529
1765	86	1962
4013	86	3071
1765	14	809
6722	14	734
6386	61	3729
1765	61	3110
6722	61	2738

Table 4: The `ProductSale` instance.

2.2 Explanation

The original table uses the schema:

```
SalesRecords(SalespersonNumber, ProductNumber, SalespersonName,
    ↪ CommissionPercentage, YearOfHire, DepartmentNumber, ManagerName,
    ↪ ProductName, UnitPrice, Quantity)
Primary Key: SalespersonNumber
```

I began by getting this table into First Normal Form. This requires removing the repeating groups that each row has relating to the product. I did this by creating a new table `Product`, and storing a foreign key to the `Salesperson` in the `Product`. The schema for the tables in First Normal Form is:

```
Salesperson(SalespersonNumber, SalespersonName, CommissionPercentage,
    ↪ YearOfHire, DepartmentNumber, ManagerName)
Primary Key: SalespersonNumber

Product(ProductNumber, SalespersonNumber, ProductName, UnitPrice, Quantity)
Primary Key: ProductNumber, SalespersonNumber
    Alternate Key: ProductName, SalespersonNumber
Foreign Key: SalespersonNumber references Salesperson(SalespersonNumber)
```

To get the tables into Second Normal Form, all partial dependencies must be removed. A partial dependency exists in the `Product` table, as `Quantity` only depends on `SalespersonNumber`, and not on the entire primary key (and the opposite is true for the other attributes). I therefore created a new `ProductSale` table, resulting in the following schema:

```
Salesperson(SalespersonNumber, SalespersonName, CommissionPercentage,
    ↪ YearOfHire, DepartmentNumber, ManagerName)
Primary Key: SalespersonNumber

Product(ProductNumber, ProductName, UnitPrice)
Primary Key: ProductNumber

ProductSale(SalespersonNumber, ProductNumber, Quantity)
Primary Key: SalespersonNumber, ProductNumber
Foreign Key: SalespersonNumber references Salesperson(SalespersonNumber),
    ↪ ProductNumber references Product(ProductNumber)
```

Finally, to get the tables into Third Normal Form, all transitive dependencies must be removed. Transitive dependecies existed in the `Salesperson` table, as `ManagerName` depends only on `DepartmentNumber`, which is not part of the primary key. I therefore created a new `Department` table, resulting in the following schema:

```
Salesperson(SalespersonNumber, SalespersonName, CommissionPercentage,
    ↪ YearOfHire, DepartmentNumber)
Primary Key: SalespersonNumber
Foreign Key: DepartmentNumber references Department(DepartmentNumber)

Department(DepartmentNumber, ManagerName)
Primary Key: DepartmentNumber

Product(ProductNumber, ProductName, UnitPrice)
Primary Key: ProductNumber

ProductSale(SalespersonNumber, ProductNumber, Quantity)
Primary Key: SalespersonNumber, ProductNumber
Foreign Key: SalespersonNumber references Salesperson(SalespersonNumber),
    ↪ ProductNumber references Product(ProductNumber)
```