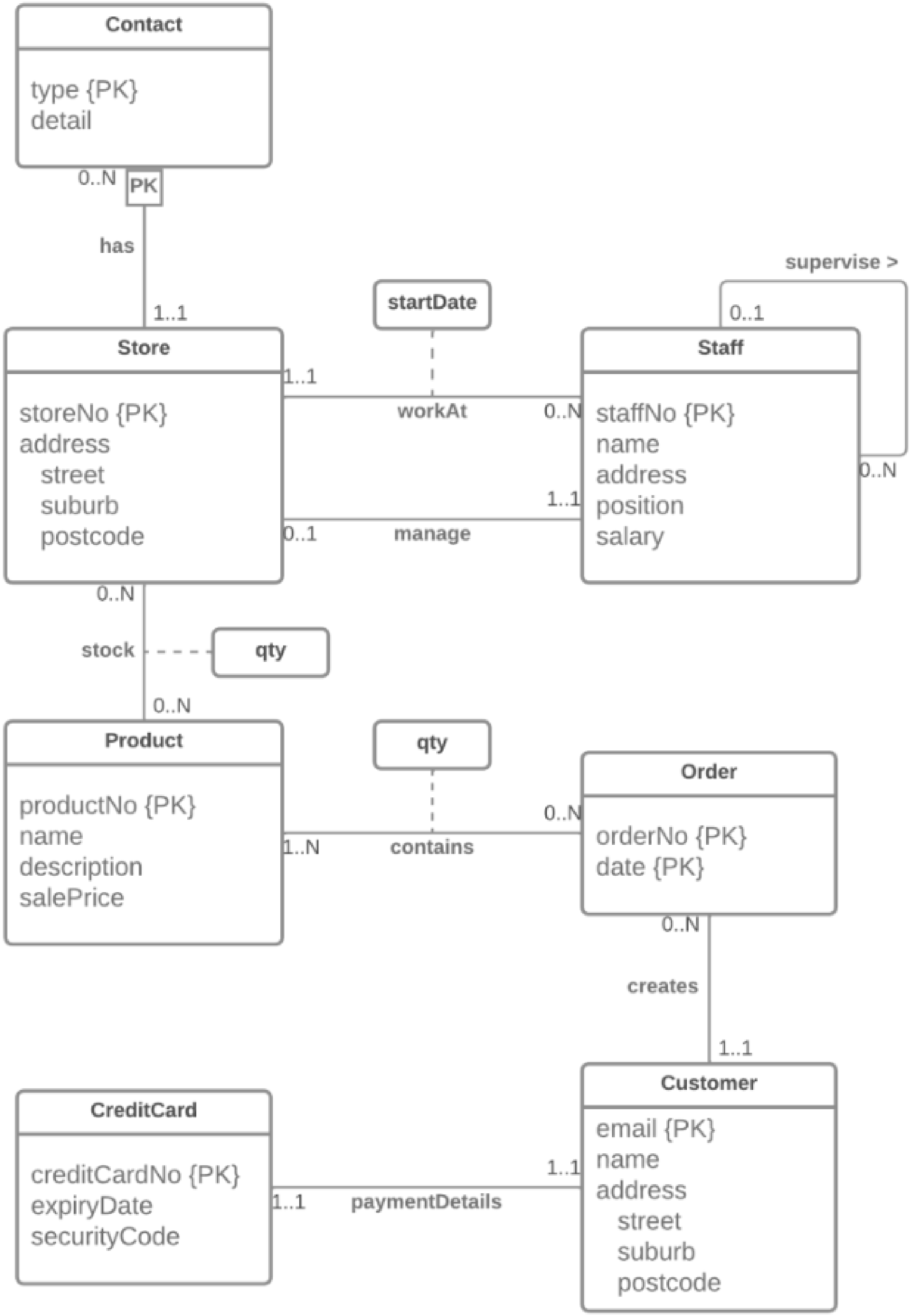
ISYS_Databases_Concepts_Assessment2

Name: Mrwan Alhandi

ID: 3969393

1databases_projects



As a result of **incorrect** application of 7-step mapping process, it was mapped into following relations.

Contact(type, StoreNo*, details, S_Postcode*)
Store(StoreNo, S_Street, S_Suburb, S_Postcode, staffNo*, startDate, ManagerStaffNo*, S_Name*)
Staff((StaffNo, S_Name, S_Address, Position, Salary, SupervisorStaffNo*, SupervisorStaffName*)
Product(ProductNo, P_Name, P_Description, P_salePrice)
Order(OrderNo, OrderDate, C_Email*, C_name*)
Customer(C_Email, C_Name, C_Street, C_Suburb,C_Postcode, CreditCardNo*, expiryDate*, securityCode*)

Contains(productNo*, OrderNo*, OrderDate*, Contains_qty)
Stock(StoreNo*, ProductNo*, P_Name*, P_salePrice*, stock_qty)

Part B: SQL (40 Marks)

LibraryDB is a database system that keeps track of information concerning the books and their circulation in an imaginary library.

Disclaimer: The data that populates the database are artificially constructed and by no means correspond to actual real-world data. The schema for the LibraryDB database is given below.

borrow(transactionID, personID*, borrowdate, duedate, returndate)

author(authorID, firstname, middlename, lastname)

book_copy(bookID, bookdescID*)

book(bookdescID, title, subtitle, edition, voltitle, volnumber, language, place, year, isbn, dewey,subjectID*)

borrow_copy(transactionID*, bookID*)

person(personID, firstname, middlename, lastname, address, city, zipcode, phonenumber, emailaddress, classification, studentno, idcardno)

publisher(publisherID, publisherfullname)

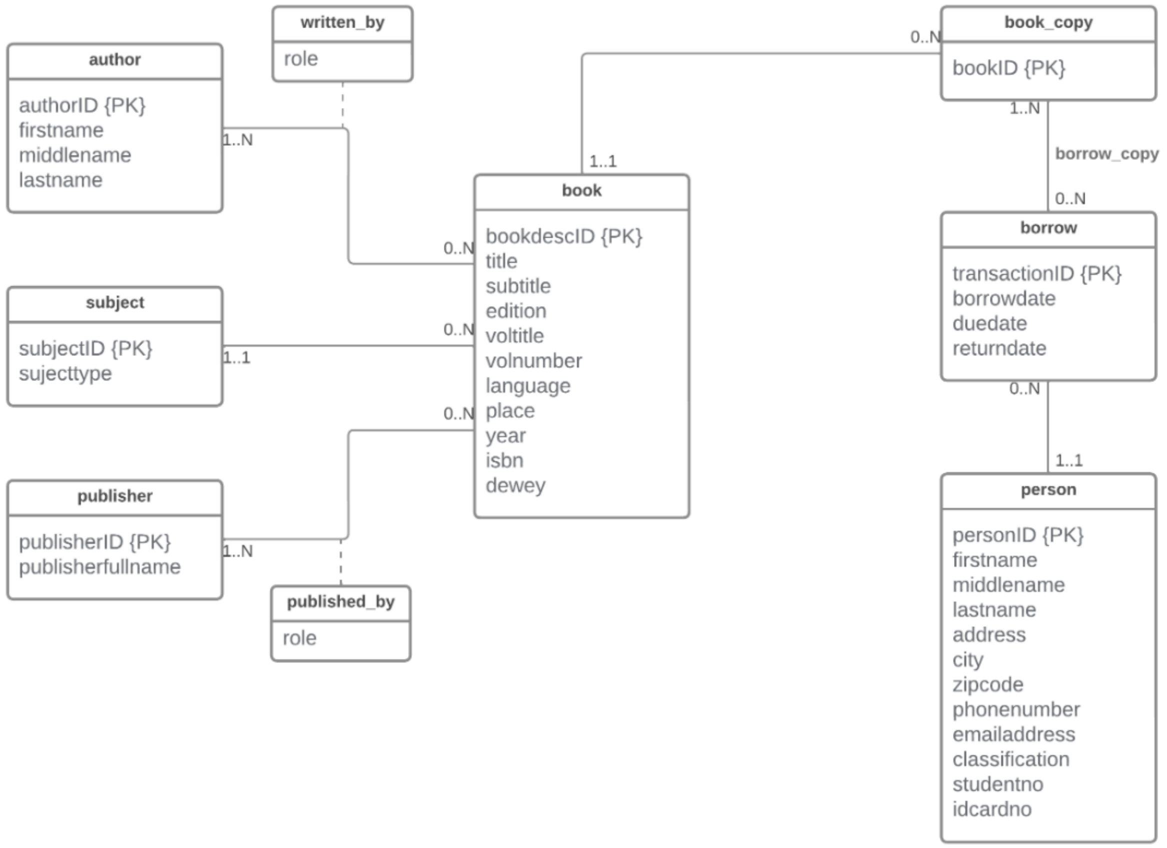
written_by(bookdescID*, authorID*, role)

published_by(bookdescID*, publisherID*, role)

subject(subjectID, subjecttype)

The primary keys are underlined. The foreign keys are denoted by asterisks (*).

A conceptual data model (shown as an entity-relationship diagram) which represents these data is given below.



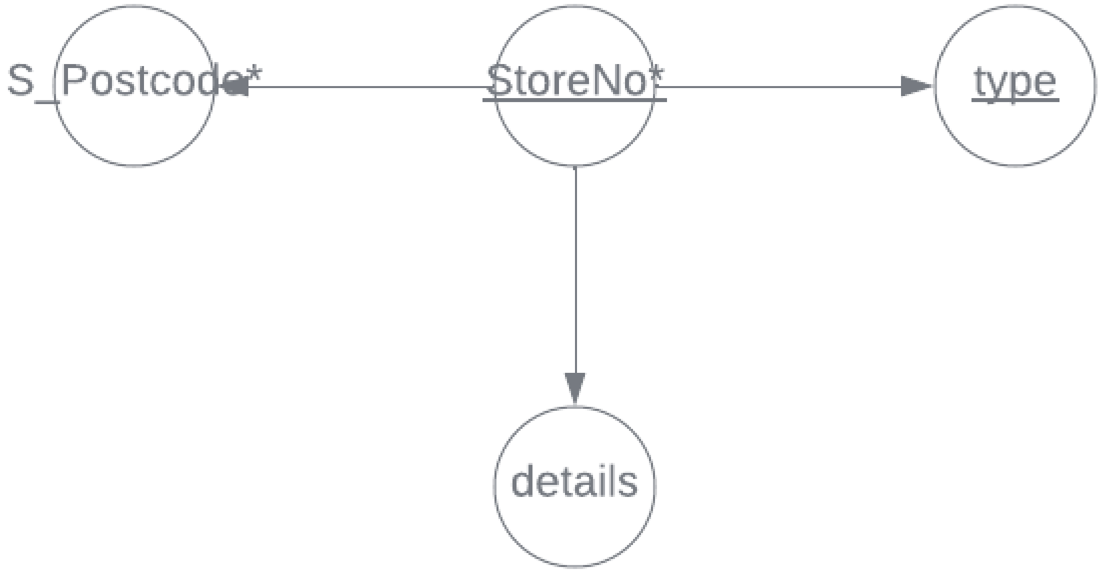
Part A: Relational Database Design

1. For each of these relations, write down all functional dependencies. If there are no functional dependencies among attributes, you must state so. Do not write down trivial functional dependencies, such as Email --> Email.

Contact

Relation:

Contact(type,StoreNo*,details,S_Postcode*)



Functional dependecies:

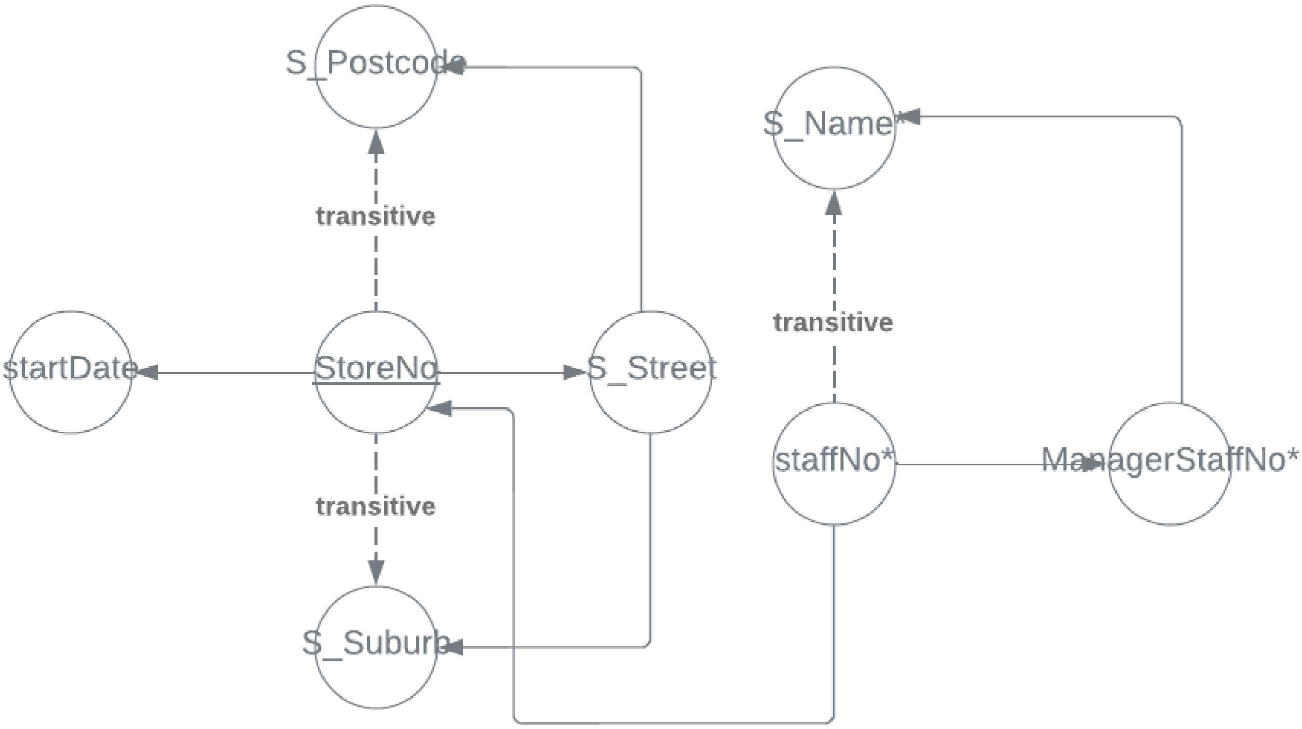
StoreNo --> type,details,S_Postcode*

Primary Key based on functional dependecies:

StoreNo

Store

Relation:
Store(StoreNo,S_Street,S_Suburb,
S_Postcode,staffNo*,startDate,ManagerStaffNo*,S_Name*)



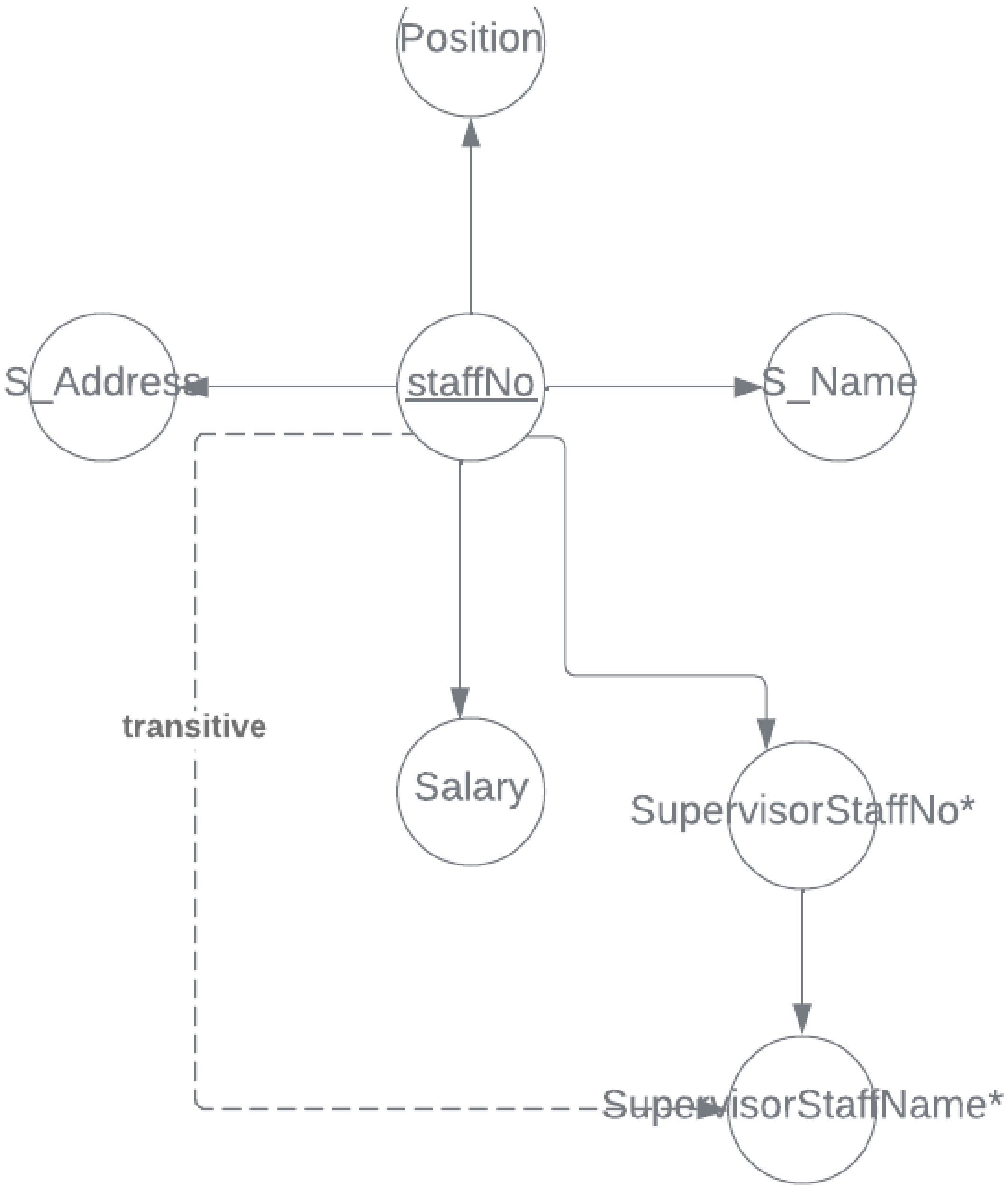
Assumption made:
Each staff can work at only one store.

Functional dependecies:
StoreNo --> startDate,S_Postcode,S_Suburb,S_Street,staffNo*
staffNo* --> ManagerStaffNo*,S_Name
ManagerStaffNo* --> S_Name
S_Street --> S_Suburb,S_Postcode
staffNo* --> StoreNo

Primary Key based on functional dependecies (composite key):
StoreNo,staffNo

Staff

Relation:
Staff(staffNo, S_Name, S_Address, Position,
Salary, SupervisorStaffNo*,
SupervisorStaffName*)



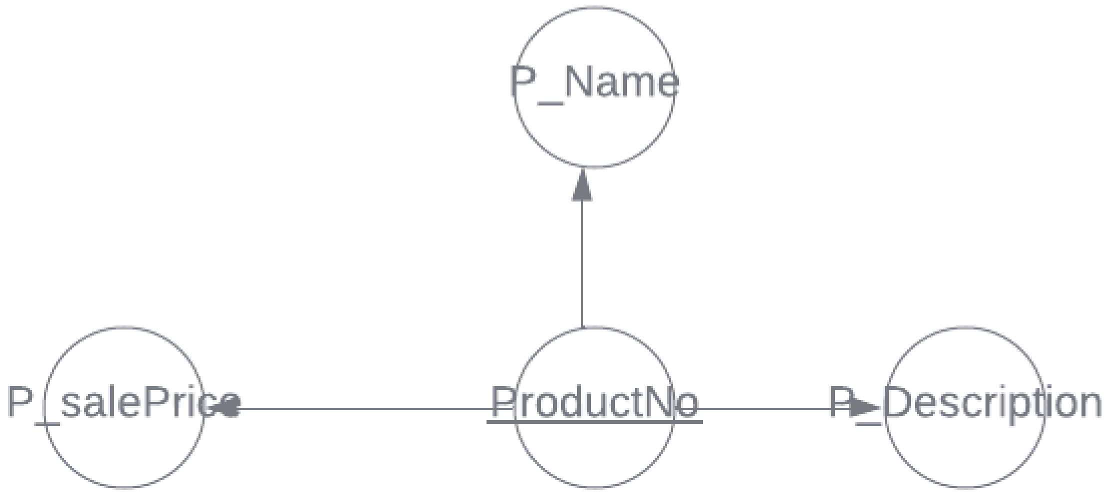
Assumption made:
Each staff have one supervisor. If the staff is supervisor, a value of null for attributes supervisorStaffNo and SupervisorStaffName* should be provided.

Functional dependecies:
staffNo --> SupervisorStaffNo*,Salary,Position,S_Name,S_Address
SupervisorStaffNo* --> SupervisorStaffName*
staffNo --> SupervisorStaffName*

Primary Key based on functional dependecies:
staffNo

Product

Relation:
Product(ProductNo, P_Name, P_Description, P_salePrice)



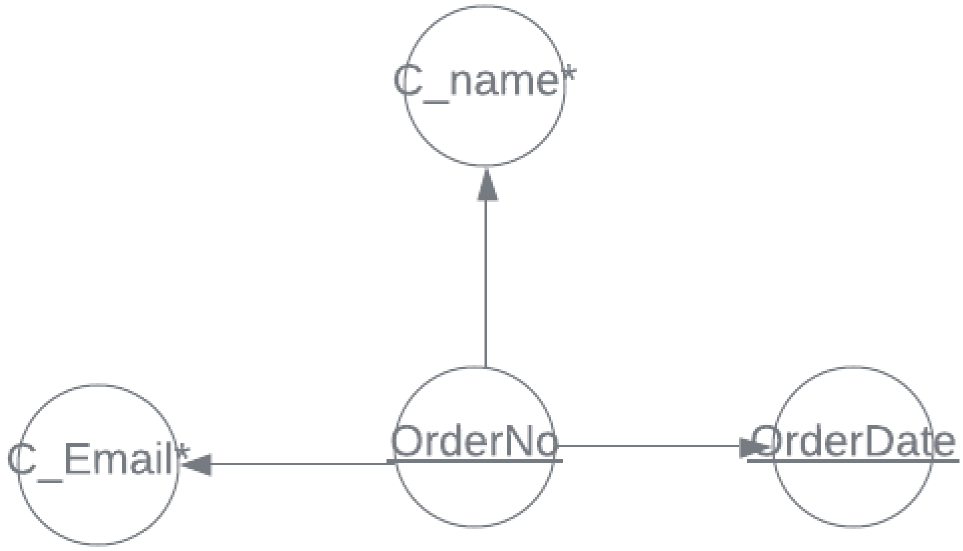
Functional dependecies:
ProductNo --> P_salePrice,P_Name,P_Description

Primary Key based on functional dependecies:
ProductNo

Order

Relation:

Order(OrderNo, OrderDate, C_Email*, C_name*)



Functional dependecies:

OrderNo --> OrderDate,C_name*,C_Email*

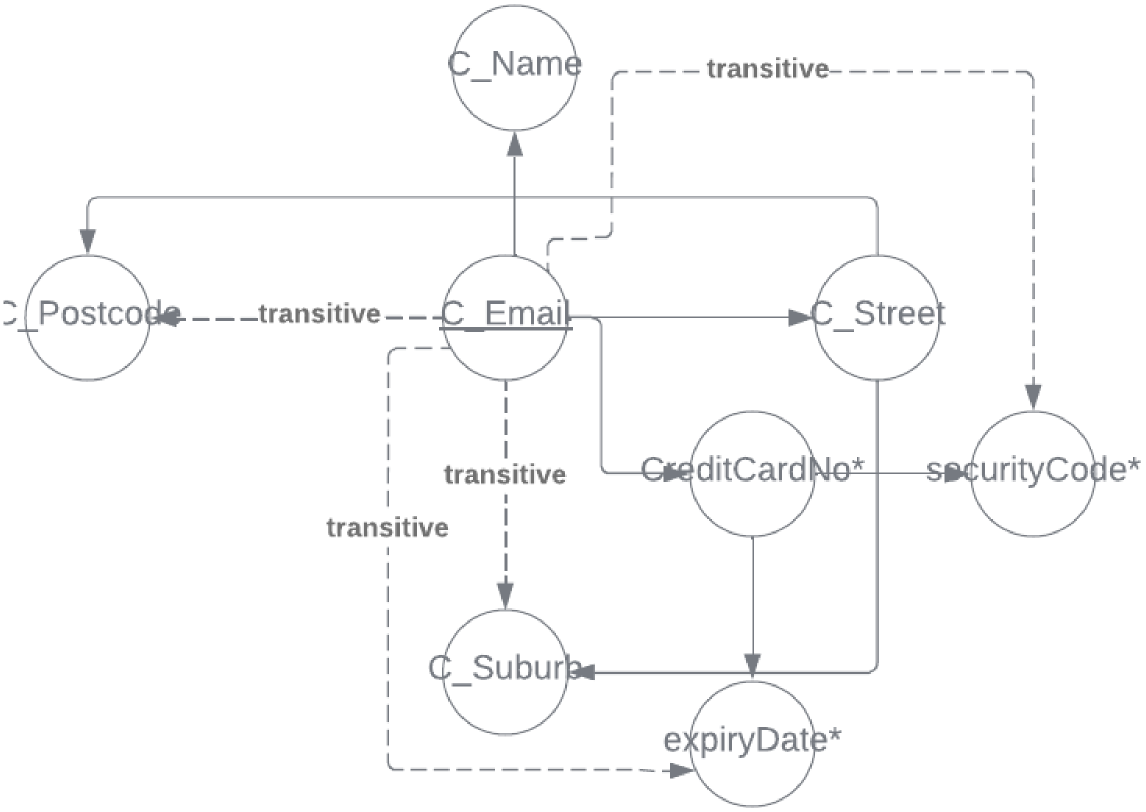
Primary Key based on functional dependecies:

OrderNo

Customer

Relation:

Customer(C_Email, C_Name, C_Street,
C_Suburb,C_Postcode, CreditCardNo*,
expiryDate*,securityCode*)



Functional dependecies:

C_Email --> C_Name,C_Street,C_Postcode,C_Postcode,CreditCardNo*,expiryDate*
CreditCardNo* --> expiryDate*,securityCode*
C_Email --> securityCode*
C_Email --> expiryDate*

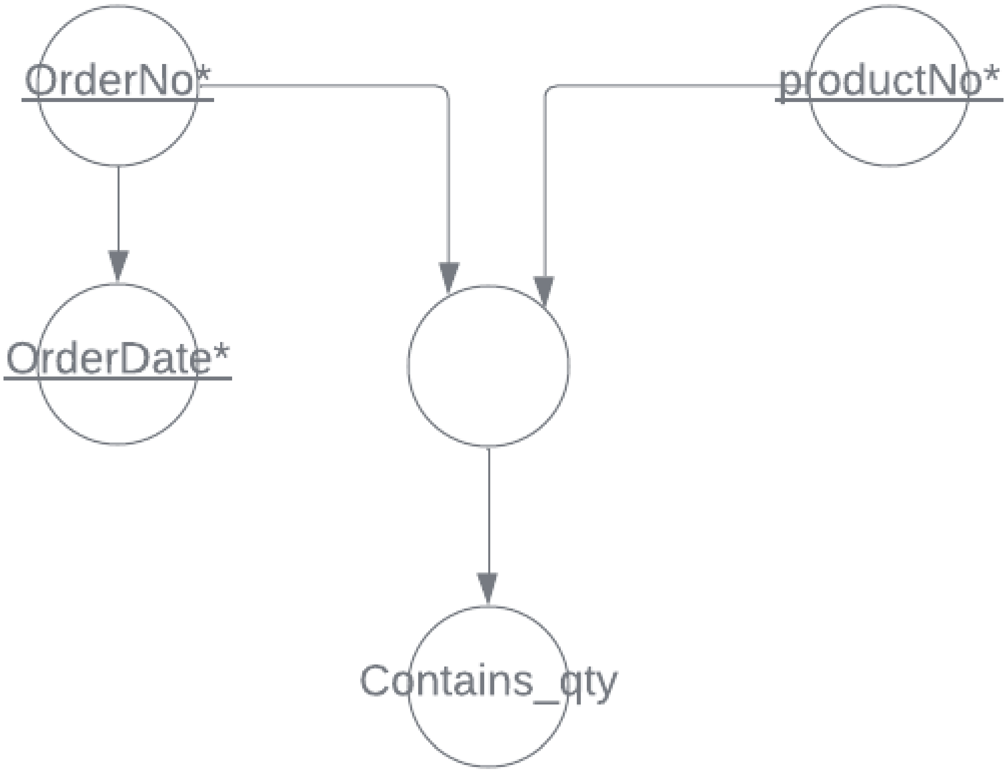
Primary Key based on functional dependecies:

C_Email

Contains

Relation:

Contains(productNo*, OrderNo*, OrderDate*,
Contains_qty)



Functional dependecies:

OrderNo* --> OrderDate*
OrderNo*,productNo* --> Contains_qty

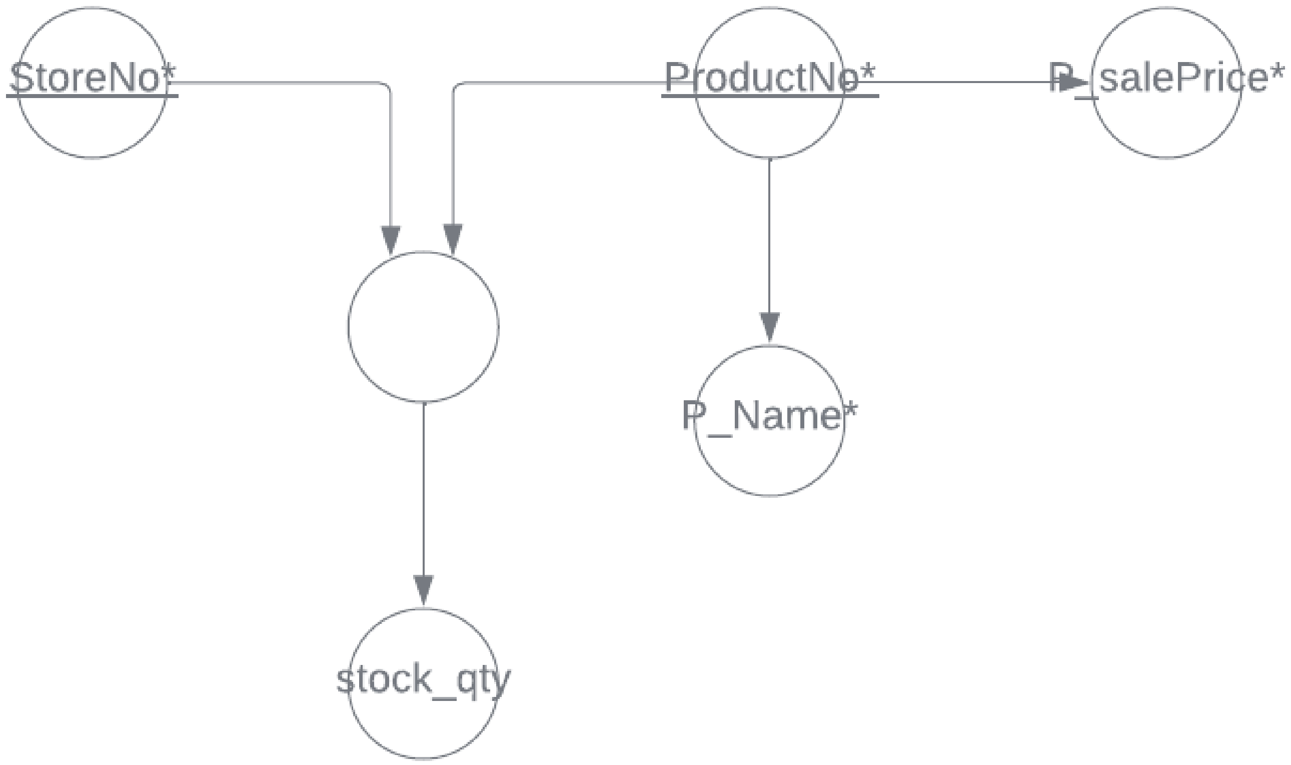
Primary Key based on functional dependecies (composite key):

OrderNo,productNo

Stock

Relation:

Stock(StoreNo*, ProductNo*, P_Name*, P_salePrice*, stock_qty)



Functional dependecies:

ProductNo* --> P_salePrice*,P_Name*

ProductNo*,StoreNo* --> stock_qty

Primary Key based on functional dependecies (composite key):

ProductNo*,StoreNo*

2. Write down the highest normal form each of these relations are in. For each of these relations, state the reasons why it doesn't meet the next normal form requirements. This is not required if the relation is in 3NF.

- A relation is in 1NF if every attribute depends on the primary key.
- A relation is in 2NF if it is already in 1NF and every attribute must be dependent on the whole key (composite) if exists. Any attribute that does not, it must be moved to a different table.
- A relation is in 3NF if it is already in 2NF and there are no transitive dependencies (attribute that is not a key that depends on an attribute that is also not a key).

Contact

- All attribute depends on a primary key. 1NF pass.
- There is no composite key. 2NF pass.
- There are no any transitive dependency. 3NF pass

Store

- All relation depends on a primary key. 1NF pass.

- The relation have a composite key which is StoreNo,staffNo. The following attributes do not depends on the composite key: S_Name, ManagerStaffNo, S_Street, S_Suburb, startDate, and S_Postcode. Hence, the relation is not in 2NF.
- There is a transitive dependency because of the attribute S_Name,S_Postcode,S_Street and S_Suburb.

Staff

- All attribute depends on a primary key. 1NF pass.
- There is no composite key. 2NF pass.
- There is a transitive dependency because of the attribute SupervisorStaffName*. Therefore, Staff is not in 3NF.

Product

- All attribute depends on a primary key. 1NF pass.
- There is no composite key. 2NF pass.
- There are no any transitive dependency. 3NF pass

Order

- All attribute depends on a primary key. 1NF pass.
- There is no composite key. 2NF pass.
- There are no any transitive dependency. 3NF pass

Customer

- All attribute depends on a primary key. 1NF pass.
- There is no composite key. 2NF pass.
- Transitive dependecies because of: C_Postcode,C_Suburb,expiryDate,*securityCode*.

Contains

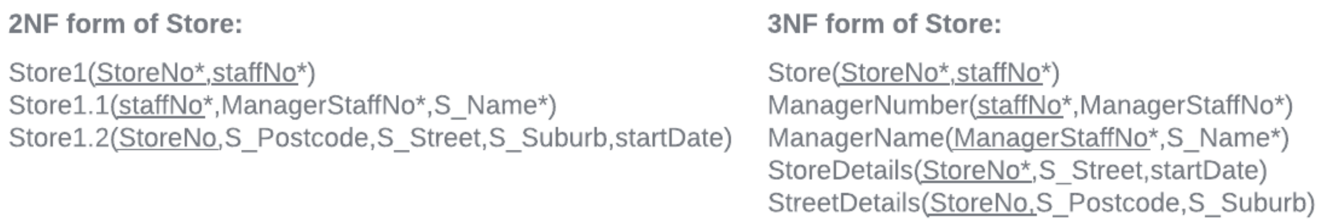
- All attribute depends on a primary key. 1NF pass.
- The relation have a composite key which is OrderNo,productNo. The following attributes do not depends on the composite key: OrderDate*. Hence, the relation is not in 2NF.
- There is no any transitive dependecies. 3NF pass after 2NF pass.

Stock

- All attribute depends on a primary key. 1NF pass.
- The relation have a composite key which is ProductNo,StoreNo. The following attributes do not depends on the composite key: OrderDate*. Hence, the relation is not in 2NF.
- There is no any transitive dependecies. 3NF pass after 2NF pass.

3. If they are not in 3NF, decompose them into 3NF relations. Write down the full database schema at the end of this step, eliminating decomposed relations and replacing them with newly created relations.

Store



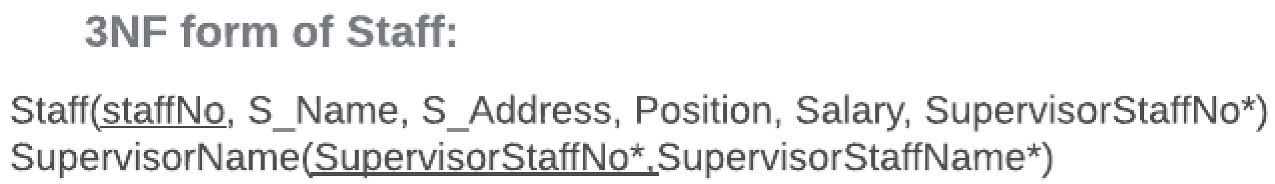
2NF form of Store:

Store1(StoreNo*,staffNo*)
Store1.1(staffNo*,ManagerStaffNo*,S_Name*)
Store1.2(StoreNo,S_Postcode,S_Street,S_Suburb,startDate)

3NF form of Store:

Store(StoreNo*,staffNo*)
ManagerNumber(staffNo*,ManagerStaffNo*)
ManagerName(ManagerStaffNo*,S_Name*)
StoreDetails(StoreNo*,S_Street,startDate)
StreetDetails(StoreNo,S_Postcode,S_Suburb)

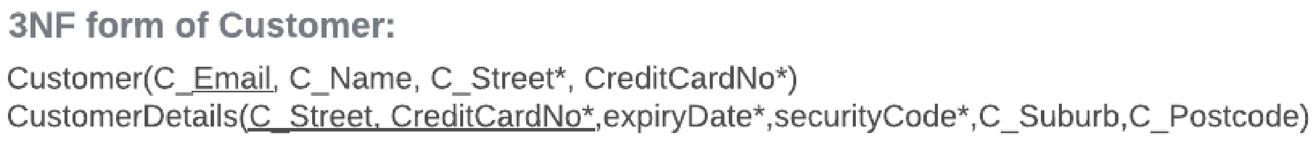
Staff



3NF form of Staff:

Staff(staffNo, S_Name, S_Address, Position, Salary, SupervisorStaffNo*)
SupervisorName(SupervisorStaffNo*,SupervisorStaffName*)

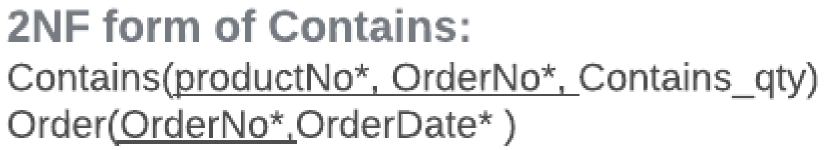
Customer



3NF form of Customer:

Customer(C_Email, C_Name, C_Street*, CreditCardNo*)
CustomerDetails(C_Street, CreditCardNo*,expiryDate*,securityCode*,C_Suburb,C_Postcode)

Contains

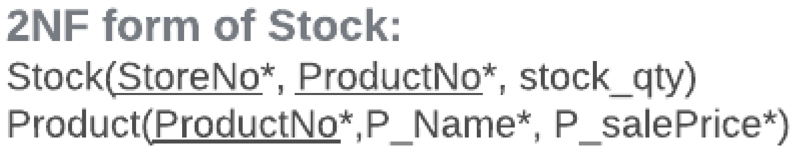


2NF form of Contains:

Contains(productNo*, OrderNo*, Contains_qty)
Order(OrderNo*,OrderDate*)

- The relation do not have any transitive dependency. Hence, the relation is in 3NF.

Stock



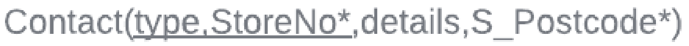
2NF form of Stock:

Stock(StoreNo*, ProductNo*, stock_qty)
Product(ProductNo*,P_Name*, P_salePrice*)

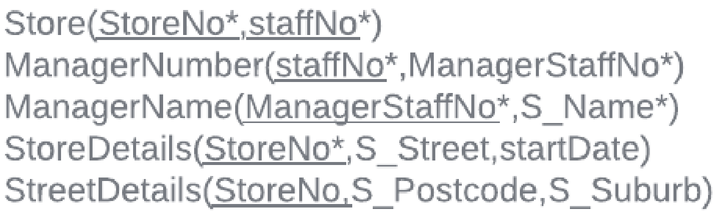
- The relation do not have any transitive dependency. Hence, the relation is in 3NF.

4. Where possible, combine the relations resulting from Part 3. Write down the full database schema at the end of this step, eliminating combined relations and replacing them with newly created relations.

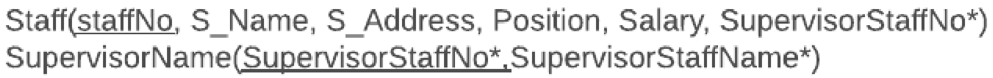
- The following are all the relations created based on the normolization process:



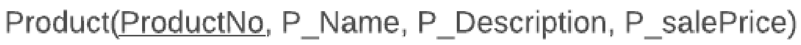
Contact(type,StoreNo*,details,S_Postcode*)



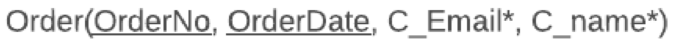
Store(StoreNo*,staffNo*)
ManagerNumber(staffNo*,ManagerStaffNo*)
ManagerName(ManagerStaffNo*,S_Name*)
StoreDetails(StoreNo*,S_Street,startDate)
StreetDetails(StoreNo,S_Postcode,S_Suburb)



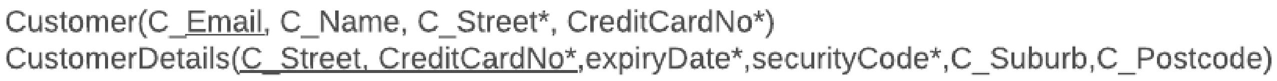
Staff(staffNo, S_Name, S_Address, Position, Salary, SupervisorStaffNo*)
SupervisorName(SupervisorStaffNo*,SupervisorStaffName*)



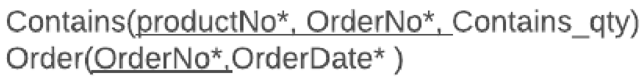
Product(ProductNo, P_Name, P_Description, P_salePrice)



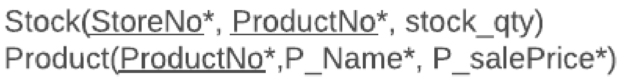
Order(OrderNo, OrderDate, C_Email*, C_name*)



Customer(C_Email, C_Name, C_Street*, CreditCardNo*)
CustomerDetails(C_Street, CreditCardNo*,expiryDate*,securityCode*,C_Suburb,C_Postcode)



Contains(productNo*, OrderNo*, Contains_qty)
Order(OrderNo*,OrderDate*)



Stock(StoreNo*, ProductNo*, stock_qty)
Product(ProductNo*,P_Name*, P_salePrice*)

- There is no need for the relation ManagerNumber. I added ManagerStaffNo* to Staff Relation.

- I removed the foreign key from StoreNo in StoreDetails.

- I removed the foreign key of S_Name attribute in ManagerName.

- There are two Product relations. I removed the second one since it have wrong foreign keys and does not have P_Description attribute.

- There are two Order relations. I removed the secnd one since its missing the foreign key of Customer. I fixed the first one primary key (OrderDate should not be a primary key). Also, I removed the foreign key of C_Name since C_Email is the primary key of Customer relation.

- I removed the foreign key from expiryDate and securityCode in CustomerDetails relation.

- I removed the foreign key from CreditCardNo from CustomerDetails.

5. Write down the final relational database schema.

Store(StoreNo*,staffNo*)

StoreDetails(StoreNo,S_Street,startDate)

StreetDetails(StoreNo,S_Postcode,S_Suburb)

Staff(staffNo, ManagerStaffNumber*,SupervisorStaffNo*,S_Name,
S_Address, Position, Salary)

ManagerName(ManagerStaffNo*,S_Name)

SupervisorName(SupervisorStaffNo*,SupervisorStaffName*)

Product(ProductNo, P_Name, P_Description, P_salePrice)

Order(OrderNo, C_Email*,OrderDate, C_Name)

Customer(C_Email,C_Street*, CreditCardNo*,C_Name)

CustomerDetails(C_Street, CreditCardNo,expiryDate,securityCode,C_Suburb,C_Postcode)

Contains(productNo*, OrderNo*, Contains_qty)

Stock(StoreNo*, ProductNo*, stock_qty)

Part B: SQL

- LibraryDB is a database system that keeps track of information concerning the books and their circulation in an imaginary library.

Question 1:

```
SELECT
        *
FROM
        person
WHERE zipcode = 11147;
```

Question 2 (a):

```
SELECT
        a.firstname,a.lastname
FROM
        author a
WHERE a.authorid IN (
        SELECT
                w.authorid
        FROM
                written_by w
        WHERE w.role = 'Translator');
```

Question 2 (b):

```
SELECT
        DISTINCT a.firstname,a.lastname
FROM
        author a
INNER JOIN written_by w USING(authorid)
WHERE w.role = 'Translator';
```

Question 3:

```
SELECT
        a.firstname,a.middlename,a.lastname,b.title
FROM
```

```
        author a
INNER JOIN written_by w USING(authorid)
INNER JOIN book b USING(bookdescid)
WHERE b.title LIKE '%Computing%';
```

Question 4:

```
SELECT
        DISTINCT b.title
FROM
        book b o
NATURAL JOIN borrow_copy
NATURAL JOIN book_copy;
```

Question 5 (a):

```
SELECT
        b.title
FROM
        book b
WHERE b.title LIKE '%DATABASE%' AND b.title NOT LIKE '%AN INTRODUCTION TO DATABASE SYSTEMS%'
EXCEPT
SELECT
        bc.bookid
FROM
        borrow_copy bc;
```

Question 5 (b):

```
SELECT
        b.title
FROM
        book b
INNER JOIN written_by wb USING(bookdescid)
WHERE wb.role = 'Author' AND b.title LIKE '%DATABASE%' AND b.title NOT LIKE 'AN INTRODUCTION TO DATABASE SYSTEMS';
```

Question 6:

```
SELECT
        p.publisherfullname
FROM
        publisher p
INNER JOIN published_by pb USING(publisherid)
INNER JOIN book b USING(bookdescid)
WHERE pb.role = 'Editor' AND b.subjectid IN (
        SELECT
                s.subjectid
        FROM
                subject s
        WHERE subjecttype = 'Networks');
```

Question 7 (a):

```
SELECT
        DISTINCT p.personid,p.firstname,p.middlename,p.lastname
FROM
        person p
        LEFT OUTER JOIN borrow b USING(personid)
WHERE p.personid NOT IN (
SELECT
        b.personid
FROM
        borrow b
);
```

Question 7 (b):

```
SELECT
        p.personid,p.firstname,p.lastname
FROM
        person p
WHERE p.personid NOT IN (
        SELECT
                DISTINCT p.personid
        FROM
                person p
        INNER JOIN borrow b USING(personid));
```

Question 8

```
SELECT
        a.firstname,a.middlename,a.lastname,a.authorid
FROM
        author a
WHERE a.firstname NOT LIKE '%Sidney%' AND a.lastname NOT LIKE '%Soclof%' AND a.authorid IN (
SELECT
        wb.authorid
FROM
        written_by wb
WHERE wb.role = 'Author'
        );
```

Question 9

```
SELECT
        p.firstname,p.lastname, COUNT(*) "Num Borrowed"
FROM
        person p
INNER JOIN borrow b USING(personid)
INNER JOIN borrow_copy bc USING(transactionid)
GROUP BY b.personid
HAVING COUNT(*) > 3
ORDER BY "Num Borrowed" desc;
```

Question 10

```
SELECT
*
FROM
person p
INNER JOIN borrow b USING(personid)
INNER JOIN borrow_copy USING(transactionid);
SELECT
bc.bookdescid
FROM
book_copy bc
GROUP BY bc.bookdescid
HAVING COUNT(bc.bookdescid) > 1

/*I did not know how to do it.*/
```

Question 11

a) The database can already handle loan extensions. This is because there is duedate attribute in borrow entity. The user can simply ask the borrower for the transactionID. Using a simply query that include where statement, the duedate can be updated. Also, the returndate must be updated.

Assume that a user with a transactionID = 8 want to extend the loan:

```
UPDATE borrow
SET duedate = 1
WHERE transactionID = 8;
UPDATE borrow
SET returndate = CASE
WHEN returndate < duedate THEN NULL
ELSE returndate
```

```
END
WHERE transactionID = 8;
```

b) We can create a new attribute E.g loanCounter in borrow entity that counts how many times the customer have extended their loan. This attribute have a default value of 0. By using an if statement we can set a condition that it is only possible to update the due date if the new attribute E.g loanCounter is < 2.

```
ALTER TABLE borrow
ADD loanCounter INTEGER DEFAULT 0;

SELECT IF(b.loanCounter < 2 ,UPDATE borrow b
SET b.duedate = 1
WHERE b.transactionID = 8, "You can't extend the loan more than two times.")
FROM borrow b;

/*I did not know how to do it.*/
```

Part C: Research Question

Factors involved when choosing DBMS

There are many factors that can be considered when choosing a DBMS some of them are: control of data redundancy, data consistency, the ability to have more information from the same amount of data, sharing of data, capabilities of data integrity, the security from unauthorized users, the ability to add or enforce the standards, increase/decrease in cost savings, balance of conflicting requirements, data accessibility and responsiveness, increase/decrease in productivity, maintance, concurrency, backup and recovery, complexity, size, cost of DBMS, additional hardware costs, cost of conversion, performance, and how big the impact is if the DBMS failed (Connolly and Begg 2014). Choosing which factors to consider when choosing a DBMS is dependent on the database development life cycle. It includes eight steps which are planning, requirement gathering, conceptual design, logical design, physical design, construction, implementation and rollout and ongoing support (Wilbert 2017). These steps specially the first two will help the database expert deciding which factors are crucial to consider.

SQL vs NoSQL?

It is obviously the case to use a DBMS that supports SQL rather than NoSQL since the data that is beign recorded are exteremely relational. For example, making a table for the user information that include the following attributes: the name of the user registered, date of birth, Individual Healthcare Identifier, descent, primary spoken language, residential postal address.. etc. By making for example the Individual Healthcare Identifier the primary key for the table, the access of each user information can be easily done by the usage of CRUD that SQL provide. Then, by making the Individual Healthcare Identifier a foreign key for another created table that include all the information about vaccination as an example, an easy access to each user vaccination can be done and so on. NoSQL are for non-relational and it is for key-value,graph, or wide-column stores. It is best used for unstructured data such as documents or JSON (SQL vs NoSQL: 5 Critical Differences 23 July 2021).

What to consider for a Medicare app?

A medicare app will record sensative data from users . A data breach to these type of data can effect millions of users and many losts to the organization or company. As an example, a recent data breach occured to the second largest telecommunications provider Optus and the hacker has obtained 10 million australians sensative details. Almost 40% of registered Optus users faces an enhanced likelihood of fraud. Therefore, information like these require high levels of security (9news 2022). Another factor to consider is costs and goverments prefer to use open source to reduce costs (MySQL n.d.). Since every citezen would probably be using the serivce and government matters must be handled as fast as possible, it is important to consider data accessibility, responsiveness and increase in productivity. Data is growing faster than ever where it is possible that about 1.7 megabytes of new information can be created every second for every human being (El-Attrash 2017). Therefore, size is important specially for goverments related data since everyone must be registered. The data registered are about health (Ohio University 2020). It is important that these data can be shared to the medical authorities in order for them to make the right decisions

The chosen DBMS must have the following:
1- Must support SQL.
2- Must protect the following: the data in the database, the DBMS, associated applications, any physical server and hardwares and finally the computing and netweork infrastructure used to access the database (IBM Cloud Education 2019).
3- Must be an open source.
4- High peformance on space allocation when they contain different amounts of records and time peformance when executing CRUD on different amounts of records (Ansari 2018).
5- Unlimited size since new information/users can always be added.

Oracle vs MySQL

Since there are many RDBMS vendors, it is complex to compare all of them so I decided to compare MySQL and Oracle. MySQL is an open source whereas Oracle is not. They both are receiving regular updates. They both supports the most common operating system windows and mac, but MySQL supports more operating system such as Android. They both supports most of the fundamental features such as referential integrity and transactions. In terms of storage, the problem with Oracle is that the max table size is 4 GB and the max DB size is up to 8PB whereas MySQL DB size is unlimited and the table size up to to 256 TB which is a huge difference. Oracle are limitied to attributes up to 1000 and MySQL up to 4096. They both supports temporarly table view which is important when the data is large, but MySQL does not support Materialized view which is important when the database is large but it can be dealt with easily by saving the queries in a text file for example. MySQL does not support the intersect and except statements whereas Oracle capabilities in terms of statements and table joins are fully supported. MySQL supports more data types than Oracle. In terms of security, Oracle supports more options. For example, MySQL does not have password complexity rules nor brute-force protection wherease Oracle does (Wikipedia 2021).

If we compare Oracle and MySQL based on the chosen factors that the DBMS must have that I decided. Orcale wins on high peformance and security. The reason why Oracle wins on high performance is that it supports features like materialized view, intersect and except statements, merging joins, parallel query, and system versioned tables. It wins on security over MySQL based on only two features which are Brute-force protection and password complexity rules. MySQL wins on budget because it is an open source and on size. Oracle has a limitied DB size and MySQL does not and this attribute by it self is huge. MySQL also have larger storage than Oracle in almost everything such as max table size, max row size, and max columns per row.

Conclusion

To conclude, if the size of the database is not the main concern nor the budget then Oracle is a better choice here than MySQL. MySQL does have many security features but not as Oracle. Finally, some of the capabilites that Oracle have such as the intersect and except statements can be managed using other methods.

Refrences

Connolly and Begg (2014) Database Systems: a Practical Approach to Design, Implementation, and Management, Global Edition, Pearson Education.

Wilbert (2017), The database development lifecycle, linkedin learning, accessed 15 October 2022. https://www.linkedin.com/learning/learning-relational-databases-2/the-database-development-lifecycle#:~:text=The%20database%20development%20life%20cycle%20includes%20eight%20steps%20that%20help,rollout%2C%20and%20finally%20ongoing%20support

Smallcombe (23 July 2021) 'SQL vs NoSQL: 5 Critical Differences', Integrate.io, accessed 15 October 2022. https://www.integrate.io/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20vertically%20scalable,data%20like%20documents%20or%20JSON

Wood (4 October 2022) 'More confusion as Optus text blitz continues, number of customers exposed revealed', 9news, accessed 15 october 2022. https://www.9news.com.au/national/optus-data-breach-update-more-than-two-million-customer-identity-details-exposed/b92b17d9-fc77-430b-94ca-21def7fea61d

MySQL (n.d.) # MySQL in Government, MySQL, 15 October 2022. https://www.mysql.com/industry/government/

El-Attrash (2017) WHAT GOVERNMENT NEEDS TO KNOW ABOUT DATA MANAGEMENT, govloop, accessed 15 October 2022. https://www.govloop.com/government-needs-know-data-management/

Ohio University (2020) Health Information Systems: Health Care for the Present and Future, Ohio university, accessed 15 October 2022. https://onlinemasters.ohio.edu/blog/health-information-systems/

IBM Cloud Education (2019) Database Security, IBM, accessed 15 October 2022. https://www.ibm.com/au-en/cloud/learn/database-security#toc-database-s-3h4XsKVC

Ansari (2018) Performance Comparison of Two Database Management Systems, accessed 15 October 2022. http://www.diva-portal.org/smash/get/diva2:1278762/FULLTEXT01.pdf

Wikipedia (2021) Comparison of relational database management systems: Revision history, Wikipedia The Free Encyclopedia, accessed 15 October 2022. https://en.wikipedia.org/w/index.php?title=Comparison_of_relational_database_management_systems&action=history&offset=&limit=500