SUKHUM BOONDECHARAK (S3940976)
ISYS1055 Database Concept: Assignment 2

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 (<u>type, StoreNo*</u>, detail, S_Postcode*)

type, StoreNo → detail

StoreNo → S_Postcode

Store (<u>StoreNo</u>, S_Street, S_Suburb, S_Postcode, staffNo*, startDate, ManagerStaffNo*, S_Name*)

StoreNo → S_Street, S_Suburb, S_Postcode, ManagerStaffNo

staffNo → S_Name, startDate, StoreNo

Staff (<u>StaffNo</u>, S_Name, S_Address, Position, Salary, SupervisorStaffNo*, SupervisorStaffName*)

StaffNo → S_Name, S_Address, Position, Salary, SupervisorStaffNo

SupervisorStaffNo → SupervisorStaffName

Product (<u>ProductNo</u>, P_Name, P_Description, P_salePrice)

ProductNo → P_Name, P_Description, P_salePrice

Order (<u>OrderNo, OrderDate</u>, C_Email*, C_name*)

OrderNo, OrderDate → C_Email

C_Email → C_name

Customer (<u>C_Email</u>, C_Name, C_Street, C_Suburb,C_Postcode, CreditCardNo*, expiryDate*, securityCode*)

C_Email → C_Name, C_Street, C_Suburb,C_Postcode, CreditCardNo

CreditCardNo → expiryDate, securityCode

Contains (<u>productNo*, OrderNo*, OrderDate*</u>, Contains_qty) productNo, OrderNo, OrderDate → Contains_qty

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

StoreNo, ProductNo → stock_qty

ProductNo → P_Name, P_SalePrice

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.**

Contact:

Correct primary key is <type, StoreNo>

No repeating groups → In 1NF

S_Postcode is partially dependent on the primary key → Not in 2NF

The highest normal form is 1NF

Store:

Correct primary key is staffNo

No repeating groups → In 1NF

Primary key is a single-valued, means automatically in 2NF

There are transitive dependencies → Not in 3NF

The highest normal form is 2NF

Staff:

Correct primary key is StaffNo

No repeating groups → In 1NF

Primary key is a single-valued, means automatically in 2NF

There are transitive dependencies → Not in 3NF

The highest normal form is 2NF

Product:

Correct primary key is ProductNo

No repeating groups → In 1NF

Primary key is a single-valued, means automatically in 2NF

No transitive dependencies → In 3NF

The highest normal form is 3NF

Order:

Correct primary key is <OrderNo, OrderDate>

No repeating groups → In 1NF

No partial dependencies → In 2NF

There are transitive dependencies → Not in 3NF

The highest normal form is 2NF

Customer:

Correct primary key is C_Email

No repeating groups → In 1NF

Primary key is a single-valued, means automatically in 2NF

There are transitive dependencies → Not in 3NF

The highest normal form is 2NF

Contains:

Correct primary key is <productNo, OrderNo, OrderDate>

No repeating groups → In 1NF

No partial dependencies → In 2NF

No transitive dependencies → In 3NF

The highest normal form is 3NF

Stock:

Correct primary key is <StoreNo, ProductNo>

No repeating groups → In 1NF

P_Name is partially dependent on the primary key → Not in 2NF

The highest normal form is 1NF

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.**

Contact:

With identified PKs, it can be written as follows:

Contact (type, StoreNo, detail, S_Postcode)

S_Postcode is partially dependent on StoreNo

Decompositions:

Contact1 (type, StoreNo, detail)

Contact2 (StoreNo, S_Postcode)

Contact1 has no partial dependencies → 2NF

Contact2's PK is a single-valued → 2NF

Both relations have no transitive dependencies → 3NF

Store:

With identified PKs, it can be written as follows:

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

S_Street, S_Suburb, S_Postcode, ManagerStaffNo are transitively dependent on StoreNo

Decompositions:

Store1 (staffNo, S_Name, startDate, StoreNo)

Store2 (StoreNo, S_Street, S_Suburb, S_Postcode, ManagerStaffNo)

Store1’s PK is a single-valued → 2NF

Store2’s PK is a single-valued → 2NF

Both relations have no transitive dependencies → 3NF

Staff:

With identified PKs, it can be written as follows:

Staff (StaffNo, S_Name, S_Address, Position, Salary, SupervisorStaffNo, SupervisorStaffName)

SupervisorStaffName is transitively dependent on SupervisorStaffNo

Decompositions:

Staff1 (StaffNo, S_Name, S_Address, Position, Salary, SupervisorStaffNo)

Staff2 (SupervisorStaffNo, SupervisorStaffName)

Staff1’s PK is a single-valued → 2NF

Staff2’s PK is a single-valued → 2NF

Both relations have no transitive dependencies → 3NF

Product:

It is already in 3NF. No decomposition required.

Order:

With identified PKs, it can be written as follows:

(OrderNo, OrderDate, C_Email, C_name)

C_name is transitively dependent on C_Email

Decomposition:

Order1 (OrderNo, OrderDate, C_Email)

Order2 (C_Email, C_name)

Order1 has no partial dependencies → 2NF

Order2's PK is a single-valued → 2NF

Both relations have no transitive dependencies → 3NF

Customer:

With identified PKs, it can be written as follows:

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

expiryDate and securityCode are transitively dependent on CreditCardNo

Decompositions:

Customer1 (C_Email, C_Name, C_Street, C_Suburb,C_Postcode, CreditCardNo)

Customer2 (CreditCardNo, expiryDate, securityCode)

Customer1's PK is a single-valued → 2NF

Customer2's PK is a single-valued → 2NF

Both relations have no transitive dependencies → 3NF

Contains:

It is already in 3NF. No decomposition required.

Stock:

With identified PKs, it can be written as follows:

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

P_Name and P_salePrice are partially dependent on ProductNo

Decompositions:

Stock1 (StoreNo, ProductNo, stock_qty)

Stock2 (ProductNo, P_Name, P_SalePrice)

Stock1 has no partial dependencies → 2NF

Stock2's PK is a single-valued → 2NF

Both relations have no transitive dependencies → 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.**

After the full decomposition, we get the following relations:

R1: Contact1 (type, StoreNo, detail)

R2: Contact2 (StoreNo, S_Postcode)

R3: Store1 (staffNo, S_Name, startDate, StoreNo)

R4: Store2 (StoreNo, S_Street, S_Suburb, S_Postcode, ManagerStaffNo)

R5: Staff1 (StaffNo, S_Name, S_Address, Position, Salary, SupervisorStaffNo)

R6: Staff2 (SupervisorStaffNo, SupervisorStaffName)

R7: Product (ProductNo, P_Name, P_Description, P_salePrice)

R8: Order1 (OrderNo, OrderDate, C_Email)

R9: Order2 (C_Email, C_name)

R10: Customer1 (C_Email, C_Name, C_Street, C_Suburb,C_Postcode, CreditCardNo)

R11: Customer2 (CreditCardNo, expiryDate, securityCode)

R12: Contains (productNo*, OrderNo*, OrderDate*, Contains_qty)

R13: Stock1 (StoreNo*, ProductNo*, stock_qty)

R14: Stock2 (ProductNo, P_Name, P_SalePrice)

Among these relations:

R2 and R4 have the same primary key and can be combined.

R3 and R5 have the same primary key and can be combined.

R7 and R14 have the same primary key and can be combined.

R9 and R10 have the same primary key and can be combined.

5. **Write down the final relational database schema.**

R1: Contact1 (type, StoreNo, detail)

~~R2: Contact2 (StoreNo, S_Postcode)~~

~~R3: Store1 (staffNo, S_Name, startDate, StoreNo)~~

R4: Store2 (StoreNo, S_Street, S_Suburb, S_Postcode, ManagerStaffNo)

R5: Staff1 (StaffNo, S_Name, S_Address, Position, Salary, SupervisorStaffNo, startDate, StoreNo*)

R6: Staff2 (SupervisorStaffNo, SupervisorStaffName)

R7: Product (ProductNo, P_Name, P_Description, P_salePrice)

R8: Order1 (OrderNo, OrderDate, C_Email)

~~R9: Order2 (C_Email, C_name)~~

R10: Customer1 (C_Email, C_Name, C_Street, C_Suburb,C_Postcode, CreditCardNo)

R11: Customer2 (CreditCardNo, expiryDate, securityCode)

R12: Contains (productNo*, OrderNo*, OrderDate*, Contains_qty)

R13: Stock1 (StoreNo, ProductNo, stock_qty)

~~R14: Stock2 (ProductNo, P_Name, P_SalePrice)~~

Finally, they can be renamed to better reflect what they stand for:

R1: Contact (type, StoreNo, detail)

R4: Store (StoreNo, S_Street, S_Suburb, S_Postcode, ManagerStaffNo)

R5: Staff (StaffNo, S_Name, S_Address, Position, Salary, SupervisorStaffNo, startDate, StoreNo*)

R6: Supervisor (SupervisorStaffNo, SupervisorStaffName)

R7: Product (ProductNo, P_Name, P_Description, P_salePrice)

R8: Order (OrderNo, OrderDate, C_Email)

R10: Customer (C_Email, C_Name, C_Street, C_Suburb,C_Postcode, CreditCardNo)

R11: CreditCard (CreditCardNo, expiryDate, securityCode)

R12: Contains (productNo*, OrderNo*, OrderDate*, Contains_qty)

R13: Stock (StoreNo, ProductNo, stock_qty)

Part B: SQL

1. Display all the fields of persons whose zipcode is 11147 in the person relation. (2 marks, 3 rows)

```
SELECT *
FROM PERSON
WHERE ZIPCODE = 11147;
```

2. Display the first name and last name of the persons who have translated any books – they have a translator role. (5 rows)

a. Write your query using a sub query. (2 marks)

```
SELECT FIRSTNAME, LASTNAME
FROM AUTHOR
WHERE AUTHORID IN (
  SELECT AUTHORID
  FROM WRITTEN_BY
  WHERE UPPER(ROLE) = "TRANSLATOR");
```

b. Write your query using JOINs. (2 marks)

```
SELECT A.FIRSTNAME, A.LASTNAME
FROM AUTHOR A
  INNER JOIN WRITTEN_BY W
     ON A.AUTHORID = W.AUTHORID
WHERE UPPER(W.ROLE) = "TRANSLATOR";
```

3. Who wrote the book related to "COMPUTING" (Contains the whole word)? Display the first name, middle names, and last name of the author, and the book title. Each author's role in the writing of the book is described in "role" attribute in written_by table. (3 marks, 8 rows)

```
SELECT A.FIRSTNAME, A.MIDDLENAME, A.LASTNAME, B.TITLE
FROM BOOK B
  INNER JOIN WRITTEN_BY W
     ON B.BOOKDESCID = W.BOOKDESCID
  INNER JOIN AUTHOR A
     ON W.AUTHORID = A.AUTHORID
WHERE UPPER(B.TITLE) LIKE "%COMPUTING%";
```

4. Display the titles of books that have been borrowed using NATURAL JOINS. (2 marks, 109 rows)

```
SELECT DISTINCT(B1.TITLE)
FROM BOOK B1
  NATURAL JOIN BOOK_COPY B2
  NATURAL JOIN BORROW B3
  NATURAL JOIN BORROW_COPY B4;
```

5. A borrower wants to borrow the book titled "AN INTRODUCTION TO DATABASE SYSTEMS", but all of its copies are already borrowed by others. Write two queries to display the title of other recommended books using the following methods.

a. Using partial matching of the book title -- note that the borrower is interested in a "DATABASE" book. You must use a set operator as part of your query. (3 marks, 2 rows)

```
SELECT TITLE
FROM BOOK
WHERE UPPER(TITLE) LIKE "%DATABASE%"
  EXCEPT
  SELECT TITLE
  FROM BOOK
  WHERE UPPER(TITLE) = "AN INTRODUCTION TO DATABASE SYSTEMS";
```

b. By searching of other books written by the same author (i.e. the author of "AN INTRODUCTION TO DATABASE SYSTEMS") (3 marks, 0 rows)

```
SELECT TITLE
FROM BOOK
WHERE BOOKDESCID = (
  SELECT W.BOOKDESCID
  FROM BOOK B
     INNER JOIN WRITTEN_BY W
        ON B.BOOKDESCID = W.BOOKDESCID
  WHERE UPPER(B.TITLE) = "AN INTRODUCTION TO DATABASE SYSTEMS" AND
UPPER(W.ROLE) = "AUTHOR")
EXCEPT
  SELECT TITLE
  FROM BOOK
  WHERE UPPER(TITLE) = "AN INTRODUCTION TO DATABASE SYSTEMS";
```

6. Display the list of editors who have edited books on the subject "Networks". Your query should display publisher's full name. (3 marks, 2 rows)

```
SELECT A.FIRSTNAME || " " || A.MIDDLENAME || " " || A.LASTNAME AS "FULLNAME",
  B.TITLE, P.PUBLISHERFULLNAME
FROM PUBLISHED_BY PB
  INNER JOIN WRITTEN_BY W
     ON PB.BOOKDESCID = W.BOOKDESCID
  INNER JOIN BOOK B
     ON PB.BOOKDESCID = B.BOOKDESCID
  INNER JOIN AUTHOR A
     ON W.AUTHORID = A.AUTHORID
  INNER JOIN SUBJECT S
     ON B.SUBJECTID = S.SUBJECTID
  INNER JOIN PUBLISHER P
     ON PB.PUBLISHERID = P.PUBLISHERID
WHERE UPPER(PB.ROLE) = "EDITOR" AND UPPER(S.SUBJECTTYPE) = "NETWORKS";
```

7. Display the id and full names of persons who have never borrowed any books. (333 rows)

a. Write your query using OUTER JOINs. (3 marks)

```
SELECT P.PERSONID, P.FIRSTNAME || " " || P.LASTNAME AS "FULLNAME"
FROM PERSON P
  LEFT OUTER JOIN BORROW B
     ON P.PERSONID = B.PERSONID
WHERE B.PERSONID IS NULL;
```

b. Write the query again without using OUTER JOINs. (3 marks)

```
SELECT PERSONID,
  FIRSTNAME || " " || LASTNAME AS "FULLNAME"
FROM PERSON
WHERE PERSONID NOT IN (
  SELECT DISTINCT(PERSONID)
  FROM BORROW);
```

8. Display full names of authors with whom the author 'Sidney Soclof' did not published his book(s). Your query must use IN (/NOT IN) clause. (3 marks, 351 rows)

```
SELECT DISTINCT(A.FIRSTNAME||" "||A.LASTNAME) AS "FULLNAME"
FROM WRITTEN_BY W
  INNER JOIN AUTHOR A
     ON W.AUTHORID = A.AUTHORID
WHERE UPPER(W.ROLE) = "AUTHOR"
AND W.BOOKDESCID NOT IN (
  SELECT B.BOOKDESCID
  FROM WRITTEN_BY W
     INNER JOIN BOOK B
        ON W.BOOKDESCID = B.BOOKDESCID
  WHERE UPPER(A.FIRSTNAME) = "SIDNEY"
  AND UPPER(A.LASTNAME) = "SOCLOF");
```

9. Display the first name and last name of persons who have borrowed more than 3 books. Along with each name, display the number of books as well (under the heading "Num Borrowed"). Order the result from most borrowed to least. (3 marks, 10 rows)

```
SELECT P.FIRSTNAME, P.LASTNAME, COUNT(B.TRANSACTIONID) AS "NUM BORROWED"
FROM BORROW B
  INNER JOIN BORROW_COPY BC
     ON B.TRANSACTIONID = BC.TRANSACTIONID
  INNER JOIN BOOK BK
     ON BC.BOOKID = BK.BOOKDESCID
  INNER JOIN PERSON P
     ON B.PERSONID = P.PERSONID
GROUP BY B.PERSONID
HAVING COUNT(B.TRANSACTIONID) > 3
ORDER BY COUNT(B.TRANSACTIONID);
```

10. List the full name of members that have borrowed different copies of the same book. Each row should list a unique pair of members. (4 marks, 15 rows)
Provide detailed answers to the following question.

```
SELECT P.FIRSTNAME || " " || P.LASTNAME AS FULLNAME, BKC.BOOKID, BK.BOOKDESCID,
BK.TITLE
FROM PERSON P
  INNER JOIN BORROW B
     ON P.PERSONID = B.PERSONID
  INNER JOIN BORROW_COPY BC
     ON B.TRANSACTIONID = BC.TRANSACTIONID
  INNER JOIN BOOK_COPY BKC
     ON BC.BOOKID = BKC.BOOKID
  INNER JOIN BOOK BK
     ON BKC.BOOKDESCID = BK.BOOKDESCID
WHERE BKC.BOOKID != BK.BOOKDESCID
ORDER BY BK.BOOKDESCID
```

This is as far as I can go. Have no idea at all how to extract those names. Have we really been trained for such a complicated query?

11. The library allows customers to extend their loans before the due date.

a. Can this database schema handle loan extensions? If so, how are they handled? If not, what changes required to the database schema? (2 marks)

Not sure if extension is a new attribute or just another verbal agreement. If it is so, then there can be an extension since the return date will only be stored when the book is returned. Or if the extension needs to be stored, one more relation, extension, will be added to store the extention_duration attribute. transactionID will be the composite primary key along with extension_duration.

b. If there is an additional condition that a customer can extend their loan only two times, can this database schema handle loan extensions? If so, how are they handled? If not, what changes required to the database schema? (2 marks)

This time let's assume it is not just a verbal agreement. Aside from the extension relation that stores extension_duration attribute, extension_time needs to be added into the new relation as well. The relationship between borrow and extension relation will be one-to-many since there can be two extensions. Extension relation will be in 3NF even though it contains partial keys. This is because there are no partial or transitive dependencies in the relation.

Part C: Research Task

In the current era of information technology, everything is at your fingertips. This metaphor includes how you manage your financial transactions, look up the daily updates, entertain your leisure time, connect to people boundlessly, as well as to keep up with your wellness.

Instead of a mountain of paper containing all of your important medical records, Australians can now locate them on their screens anywhere they want. To assist Australians in managing their Medicare at any time, the Australian government created the Express Plus Medicare mobile app. It is a digital, free service. Australian users can use it to file claims, view or save their proof of immunisation, enrol their newborns in Medicare, update their bank and contact information, manage the people listed on their Medicare card, check the balance of their Medicare support system, confirm their family, manage their decision to donate their organs, access their digital donor card, view their healthcare identifier, and locate a Medicare service centre.

A suitable backend database system for storing these data should be implemented given the volume of data that will be recorded and used by the entire nation. However, before we come to a conclusion, here is some information about the different categories of database management systems.

The category of database management systems can be divided into two primary types, the traditional relational database systems and noSQL database systems. The distinctions between the two will be briefly stated.

A database management system (DBMS) that is based on the relational paradigm is called a relational database management system (RDBMS). The way that data is both stored and displayed must be in the form of relations, or tables with relationships between them such as primary and foreign keys. A system should offer relational operators, or code that permits the testing of relationships between two entities, to change the data stored in tables. In this type of database systems, the data is set up in an organised manner. Because you always know where to find each piece of data, having your data structured with a precise structure makes working with it easier (Chiradeep, 2022).

Since the 1980s, RDBMSs have been a popular choice for data storage in databases used for financial records, manufacturing and logistical data, employee data, and other applications utilising historical, transactional data. Nevertheless, the market share is shrinking as a result of concurrency problems, scalability challenges, and the high network bandwidth needed for queries that must traverse several, highly normalised tables (Kimschmidtsbrain, 2017).

On the other hand, NoSQL database management systems are becoming more popular nowadays, in part because of their use with big data and real-time web applications (Justin, 2020). NoSQL databases were originally designed with developer productivity in mind because developers (and not storage) were starting to represent the majority of the expense of developing software (Lauren, n.d.). The actual value of adopting NoSQL databases, however, lies in their features.

Unstructured data in NoSQL databases can be stored in a variety of ways and has dynamic schemas. For your data, you can use a KeyValue store, Graph-based store, Document-oriented store, or Column-oriented store. Due to its adaptability, this leads to the fact that documents can be created without first defining their structure, every document may have a different structure, and the syntax may differ from one database to the other. (Mark, 2021)

Despite the fact that each form of database system appears to be useful, depending on the use cases, both database systems also have some disadvantages. Traditional relational database systems' inability to scale as your database expands is their fundamental drawback. You can use some methods, such as sharding, but putting them into practice is not simple. Meanwhile, due to its youth, the NoSQL community does not yet have the same level of maturity as the MySQL user group. Although the community is now expanding quickly, SQL database management systems like MySQL still have an advantage in terms of the number of very skilled users. In addition, lack of reporting tools for analysis and performance testing is a significant problem with NoSQL databases. As an example, Navicat Monitor for MySQL/MariaDB is one of many monitoring tools available for traditional RDBMS that may be used to assist you performance and adjust your instances. [Robert, 2019]

With that in mind, we will now look back at the source of information that will be stored and used. Individual patient information, including methods of identification and payment, healthcare access and control, and other management statistics will be stored in the databases. When patients need services from healthcare providers, databases from different healthcare platforms must quickly and consistently transfer this information amongst one another. This data-sharing technology is vital for keeping information structured because workflows frequently have standards.

But is speed the most important factor?

Express Plus Medicare will hold a lot of user personal information, so security needs to be given top priority to prevent data leaks. Information like organ donation can be extremely private, and we are unsure of how it might be put to use if it were to leak. Although businesses frequently use various types of databases to hold a number of customer and financial data, they frequently do so with outdated and inadequate default security configurations. Any kind of database can be left exposed or unprotected, but over the past few years, a number of breaches have all been focussed on one specific kind of database: "NoSQL" open-source databases. (LILY, 2017)

This leads me to the conclusion that both types of database management systems produce their own distinctive values. However, given that the NoSQL community has not yet reached the same level of development as the RDBMS user community. Despite the community's current rapid growth, RDBMS and other SQL database management systems still have an advantage in terms of the number of highly skilled users. Additionally, a major issue with NoSQL databases is the absence of reporting tools for analysis and performance testing when problems occur. Last but not least, despite the number of leak cases now being reported, it is still safer for Express Plus Medicare to use conventional relational database systems rather than NoSQL at this time.

References

Chiradeep BasuMallick. (2022, August 18). What Is DBMS (Database Management System)? Definition, Types, Properties, and Examples. Retrieved October 7, 2022, from https://www.spiceworks.com/tech/cloud/articles/database-management-systems-dbms (https://www.spiceworks.com/tech/cloud/articles/database-management-systems-dbms)

Kimschmidtsbrain. (2017, February 28). TRADITIONAL RELATIONAL DATABASE MANAGEMENT SYSTEMS. Retrieved October 7, 2022, from https://awskimschmidt.com/2017/02/28/traditional-relational-database-management-systems-chapter-3-6-in-all-aws-data-analytics-services (https://awskimschmidt.com/2017/02/28/traditional-relational-database-management-systems-chapter-3-6-in-all-aws-data-analytics-services)

Justin Stoltzfus. (2020, July 6). What is the difference between a NoSQL database and a traditional database management system? Retrieved October 7, 2022, from https://www.techopedia.com/7/31991/storage/what-is-the-difference-between-a-nosql-database-and-a-traditional-database-management-system#:~:text=In%20traditional%20relational%20databases%2C%20data,methods%2C%20or%20object%20store%20methods (https://www.techopedia.com/7/31991/storage/what-is-the-difference-between-a-nosql-database-and-a-traditional-database-management-system#:~:text=In%20traditional%20relational%20databases%2C%20data,methods%2C%20or%20object%20store%20methods)

Lauren Schaefer. (No Date Stamped). What is NoSQL? Retrieved October 7, 2022, from https://www.mongodb.com/nosql-explained#:~:text=NoSQL%20databases%20come%20in%20a,data%20and%20high%20user%20loads (https://www.mongodb.com/nosql-explained#:~:text=NoSQL%20databases%20come%20in%20a,data%20and%20high%20user%20loads)

Mark Smallcombe. (2021, July 23). SQL vs NoSQL: 5 Critical Differences. Retrieved October 7, 2022, from https://www.integrate.io/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%2