CS450/550: Database Concepts
Assignment 2
Name: PRASHANT TOMAR
UIN: 01215133

Q 1. Comment on following designs (whether the primary keys are appropriate).

(a) SALE (<u>CustomerName</u>, CustomerPhone, CustomerAddress, ItemName, SalePrice, SaleDate)
Answer: No, using 'CustomerName' as the primary key is not suitable because it doesn't provide a unique way to identify each row or record. Since multiple rows could have the same customer name, we can't rely on it to guarantee uniqueness for each row.

(b) SALE (CustomerName, CustomerPhone, CustomerAddress, <u>ItemName</u>, SalePrice, SaleDate)
Answer: No, the primary key, which is 'Item Name,' can't guarantee unique identification for each row or record, as the same item name might appear in multiple rows. Nevertheless, in our current context, this holds true, enabling us to uniquely access each record because every row contains a distinct item name.

(c) SALE (CustomerName, CustomerPhone, CustomerAddress, <u>ItemName</u>, SalePrice, <u>SaleDate</u>)
Answer: No, using the primary key of (ItemName, SaleDate) is not practical because it doesn't ensure the unique identification of each row or record. The item name could be duplicated for multiple rows on the same date, especially when an item sells multiple times within a day. Consequently, we cannot rely on (ItemName, SaleDate) to guarantee the uniqueness of a row. While this may hold true in the provided data, it's not a general rule.

(d) SALE (<u>CustomerName</u>, CustomerPhone, CustomerAddress, <u>ItemName,</u> SalePrice, SaleDate)
Answer: No, the primary key (CustomerName, ItemName) is not suitable for ensuring the unique identification of rows or records. This is because there is a

possibility that two distinct individuals with the same name could buy the same item. Therefore, we cannot consider a row as uniquely identifiable based on (CustomerName, ItemName).

Q 2. Add ID columns called CustomerID and SaleID into the SALE table. Answer the following questions. (35 points)

CustomerID	CustomerName	CustomerPhone	CustomerAddress	ItemName	SalePrice	SaleDate	SaleID
1	Anderson	425-125-8461	533 Main street, Chicago, IL, 62803	Antique Desk	3000	14-Dec-12	1
1	Anderson	425-125-8461	533 Main street, Chicago, IL, 62803	Lamp	200	17-Dec-12	2
2	Judy	231-234-1232	112 University Ave., State College, PA, 16802	Chair	1000	2-Dec-12	3
2	Judy	231-234-1232	112 University Ave., State College, PA, 16802	Coffee Table	150	2-Dec-12	4
3	Davis	131-122-9987	17 Green Street, Seattle, WA, 10592	Book Shelf	500	20-Dec-12	5

(a) List one candidate key in this relation.

Answer: We can consider using SaleID as the candidate key since it, when combined with CustomerID, uniquely identifies each record. This is because SaleID will be distinct for every sale transaction.

(b) List at least two functional dependencies. What is the determinant in each functional dependency?

Answer: Following are two functional dependency which I conclude:

1. SaleID →(CustomerID, CustomerName, CustomerPhone, CustomerAddress, ItemName, SalePrice, SaleDate).

   Dependent: SaleID

2. CustomerID →(CustomerName, CustomerPhone, CustomerAddress).

Dependent: CustomerID

NOTE: Additionally, there is another functional dependency that allows us to retrieve SalePrice based on ItemName and SaleDate.
I.e. (ItemName, SaleDate) → SalePrice
Dependent: (ItemName, SaleDate)

I included SaleDate along with ItemName because item prices can fluctuate over time. Having the SaleDate allows us to access the corresponding SalePrice for a specific date.

(c) Is the SALE relation well-formed? Why?
Answer: In the case mentioned above, we're accessing records through two dependencies. CustomerID isn't the candidate key, whereas SaleID is. Consequently, the relationship lacks solidity.

(d) If the relation is not well-formed, normalize it to a well-formed relation. In the well-formed relation, please indicate the primary key and foreign key
Answer: Following are the Normalization procedures to normalize it into well-formed relations.

a. Take the columns involved in the functional dependency and create a new relation for them.
CUSTOMER (CustomerID, CustomerName, CustomerPhone, CustomerAddress)

b. Designate the determinant of the functional dependency as the primary key of the new relation.
CUSTOMER (CustomerID, CustomerName, CustomerPhone, CustomerAddress)

c. Retain a copy of the determinant as a foreign key in the original relation.
SALE (SaleID, ItemName, SalePrice, SaleDate, *CustomerID*)

In the well-formed relation model design:

CUSTOMER (CustomerID, CustomerName, CustomerPhone, CustomerAddress)
SALE (SaleID, ItemName, SalePrice, SaleDate, *CustomerID*)

Q 3. Comment on following designs (whether the primary keys are appropriate).

(a) PURCHASE(ItemName, PurchasePrice, PurchaseDate, VendorName, VendorPhone, VendorAddress)
Answer: No, the primary key, VendorName, is unsuitable as it does not enable unique identification of rows or records. Since multiple rows can share the same vendor name, we cannot establish uniqueness solely based on the vendor name.

(b) PURCHASE(ItemName, PurchasePrice, PurchaseDate, VendorName, VendorPhone, VendorAddress)
Answer: No, the primary key (ItemName, VendorName) does not provide unique identification for the row or record, hindering our ability to access records due to the possibility of multiple vendors selling the same items to different customers.

(c) PURCHASE(ItemName, PurchasePrice, PurchaseDate, VendorName, VendorPhone, VendorAddress)
Answer: No, the primary key, which is VendorAddress, is not suitable for ensuring unique identification because the same vendor can provide multiple items to different customers. Consequently, multiple customers may share the same vendor address where they made their purchases. Therefore, (VendorAddress) alone cannot uniquely describe a row or record.

(d) PURCHASE(ItemName, PurchasePrice, PurchaseDate, VendorName, VendorPhone, VendorAddress)
Answer: No, the primary key (ItemName, PurchaseDate, VendorName) cannot ensure unique identification of the row or record, as it's possible for multiple consumers to purchase the same item on the same day from the same vendor. Hence, we cannot consider the row as unique based on the (ItemName, PurchaseDate, VendorName) fields.

Q 4. Add ID columns called VendorID and PurchaseID into the PURCHASE table. Answer the following questions. (35 points)

PurchaseID	ItemName	Purchase Price	PurchaseDate	VendorName	VendorPhone	VendorAddress	VendorID
1	Coffee Table	100	20-Nov-12	European Specialities	206-329-1920	18 Union St., San Francisco, CA, 28192	1
2	Antique Desk	2500	14-Nov-12	European Specialities	206-329-1920	18 Union St., San Francisco, CA, 28192	1
3	Crystal Lights	900	2-Nov-12	Lamp and Lights	231-129-1232	210 Broadway, Dallas, TX, 23212	2
4	Lamp	170	1-Nov-12	Lamp and Lights	231-129-1232	210 Broadway, Dallas, TX, 23212	2
5	Book Shelf	320	24-Oct-12	Antique Things	321-329-1203	190 Alley St., Miami, 96802	3
6	Chair	700	13-Nov-12	European Specialities	206-329-1920	18 Union St., San Francisco, CA, 28192	1

(a) List one candidate key in this relation.

Answer:

After incorporating PurchaseID and VendorID into the PURCHASE table, PurchaseID can serve as a unique identifier for each record. Therefore, we can establish PurchaseID as the candidate key, as it will be distinct for every purchase made.

(b) List at least two functional dependencies. What is the determinant in each functional dependency?

Answer: Following are two functional dependency which I found:

1. PurchaseID →(ItemName, PurchasePrice, PurchaseDate, VendorName, VendorPhone, VendorAddress, VendorID).

   Dependent: PurchaseID

2. VendorID →(VendorName, VendorPhone, VendorAddress).

Dependent: VendorID

NOTE: Additionally, there is another functional dependency where we can retrieve the PurchasePrice based on ItemName and PurchaseDate, which can be represented as follows:

I.e. (ItemName, PurchaseDate) → PurchasePrice

Dependent: (ItemName, PurchaseDate)

I included PurchaseDate along with ItemName because the purchase price can vary over time. Having the PurchaseDate allows us to access the corresponding PurchasePrice for a specific date.

(c) Is the SALE relation well-formed? Why?

Answer: In the scenario described, we are accessing records through two dependencies. However, it's important to note that PurchaseID serves as the candidate key, whereas VendorID does not. Consequently, the relationship is not properly established or well-formed.

(d) If the relation is not well-formed, normalize it to a well-formed relation. In the well-formed relation, please indicate the primary key and foreign key

Answer: Following are the Normalization procedures to normalize it into well-formed relations.

a. Create a new relation containing the columns involved in the functional dependency.

VENDOR (VendorID, VendorName, VendorPhone, VendorAddress).

b. Designate the determinant of the functional dependency as the primary key of the new relation.

VENDOR (VendorID, VendorName, VendorPhone, VendorAddress).

c. Retain a copy of the determinant as a foreign key in the original relation.

PURCHASE (PurchaseID, ItemName, PurchasePrice, PurchaseDate, *VendorID*).

In the well-formed relation model design:

VENDOR (VendorID, VendorName, VendorPhone, VendorAddress).
PURCHASE (PurchaseID, ItemName, PurchasePrice, PurchaseDate, *VendorID*).

Q 5. You may realize that items in SALE and PURCHASE tables are not connected. For example, for the lamp sold to customer Anderson, we did not keep the information of where this lamp was purchased. Add an additional ITEM table to remedy this problem. How would you design ITEM table and how would you modify the tables you designed in Question 2d and 4d accordingly? Discuss how you keep track of the items in both SALE and PURCHASE in this design. (10 points)

Answer:
To establish a connection between two tables, I created the ITEM table, where ItemID and ItemName serve as Primary Keys. In the PURCHASE and SALE tables, I replaced ItemName with ItemID, converting it into a Foreign Key in both tables. This modification enables us to query items purchased from vendors through the SALE table and establish a link between these two tables.

Here are the tables:

CUSTOMER (CustomerID, CustomerName, CustomerPhone, CustomerAddress)
SALE (SaleID, SalePrice, SaleDate, *CustomerID*, *ItemID*)
VENDOR (VendorID, VendorName, VendorPhone, VendorAddress)
PURCHASE (PurchaseID, PurchasePrice, PurchaseDate, *VendorID*, *ItemID*)
ITEM (ItemID, ItemName)

Additionally, to track item quantities and maintain separate batches using BatchID, we introduced a BATCH table. However, it's important to note that a potential situation of ambiguity may arise if the same item is purchased from multiple vendors. Here are all the necessary tables:

CUSTOMER (CustomerID, CustomerName, CustomerPhone, CustomerAddress)
SALE (SaleID, SalePrice, SaleDate, *CustomerID*, *ItemID*)

VENDOR (VendorID, VendorName, VendorPhone, VendorAddress)
PURCHASE (PurchaseID, PurchasePrice, PurchaseDate, *VendorID*, *ItemID*)
ITEM (ItemID, ItemName)
BATCH (BatchID, *VendorID*, *ItemID*, Quantity)

The BATCH table is designed to track item quantities and vendor-specific information, especially useful in cases where the same item is acquired from different vendors, ensuring clarity in the inventory management process.