Database Concepts-Assignment 2

Submitted By:
Deepthi Suresh
S3991481

Part A

Q1. The functional dependencies for the given relational schemas are as follows:

Customer (CustNo, CustName, Phone, Address, DateOfBirth, EmailAddress)

FD 1:CustNo → CustName, Phone, Address, DateOfBirth, EmailAddress

Account(AccNo, CustNo*, CustName*, EmpNo*)
FD 2: AccNo → CustNo, EmpNo
FD3: CustNo → CustName

Employee (EmpNo, RegisterBranchBSB, BranchAddress)
FD4: EmpNo → RegisterBranchBSB
FD5: RegisterBranchBSB → BranchAddress

AccType (TypeID, TypeName)
FD 6: TypeID → TypeName

CustomerACCType (CustNo*, TypeID*, CustName*, TypeName*)
FD7: CustNo → CustName
FD8: TypeID → TypeName

Q2. The primary and foreign keys are as below. There are no other candidate keys for the schemas.

Customer (CustNo, CustName, Phone, Address, DateOfBirth, EmailAddress)
FD 1:CustNo → CustName, Phone, Address, DateOfBirth, EmailAddress

All the attributes except CustNo is the right side of the FD and therefore CustNo is the PK. There are no other Candidate keys.
The relation is already in 3NF since all non-primary key attributes are fully functionally dependent on the primary key attributes and not transitively dependent on the primary key.

Account(AccNo, CustNo*, CustName*, EmpNo*)
FD 2: AccNo → CustNo, EmpNo
FD3: CustNo → CustName

AccNo is the only attribute not in the right side of the FD and hence must be the PK.
The relation is currently in 2NF as all non-primary key attributes are fully functionally dependent on the primary key attributes (and there is no composite PKs), however CustName a non-primary key attribute is transitively dependent on the primary key, AccNo, therefore not meeting the requirements of 3NF. Here the relationship between AccNo and CustName is redundant.

Employee (EmpNo, RegisterBranchBSB, BranchAddress)
FD4: EmpNo → RegisterBranchBSB
FD5: RegisterBranchBSB → BranchAddress

EmpNo is the only attribute not in the right side of the FD and hence must be the PK.
The relation is currently in 2NF as all non-primary key attributes are fully functionally dependent on the primary key attributes (and there is no composite PKs), however BranchAddress a non-primary key attribute is transitively dependent on the primary key, EmpNo, therefore not meeting the requirements of 3NF. Here the relationship between EmpNo and BranchAddress is redundant.

AccType (TypeID, TypeName)
FD 6: TypeID → TypeName

Here, TypeID is the PK and no candidate and foreign keys.

The relation is already in 3NF since all non-primary key attributes are fully functionally dependent on the primary key attributes and not transitively dependent on the primary key.

CustomerACCType (<u>CustNo*, TypeID*,</u> CustName*, TypeName*)
FD7: CustNo → CustName
FD8: TypeID → TypeName

CustNo and TypeID attributes are not in the right side of the FD and hence must be the PK.
The relation is currently in 1NF as all non-primary key attributes are functionally dependent on the primary key attributes, however there is a composite PK of CustNo and TypeID which means that the relation doesn't meet the requirement of 2NF of being fully dependent on the whole PK. Here CustName is only dependent on CustNo and not on TypeID, similarly TypeName is only dependent on TypeID and not on CustNo.

Q3. CustomerACCType relation is currently in 1NF and needs to decomposed to 2NF and 3NF

i)
Original schema: CustomerACCType (<u>CustNo*, TypeID*,</u> CustName*, TypeName*)
FD7: CustNo → CustName
FD8: TypeID → TypeName

Since CustName, TypeName are not fully functionally dependent on the PK, we must remove it from the initial relation and move them to a new one.

New relation:
Customer1 (<u>CustNo</u>, CustName)
AccType (<u>TypeID</u>, TypeName)
CustomerACCType(<u>CustNo*, TypeID*</u>)

All new relations above are now decomposed to 2NF as they are fully dependent on the whole PK in each case. The relation is automatically in 3NF since the non-primary key attributes are not transitively dependent on the primary key.

ii)
Original schema: Account(<u>AccNo</u>, CustNo*, CustName*, EmpNo*)
FD 2: AccNo → CustNo, EmpNo
FD3: CustNo → CustName

Since CustName is transitively dependent on the primary key, AccNo, we have to move it to a new table.

New relations
Account (<u>AccNo</u>, CustNo*, EmpNo*)
Customer2(<u>CustNo</u>, CustName)

All new relations above are now decomposed to 3NF as they are fully dependent on the whole PK in each case and not transitively dependent on the primary key.

iii)
Original schema: Employee (<u>EmpNo</u>, RegisterBranchBSB, BranchAddress)
FD4: EmpNo → RegisterBranchBSB
FD5: RegisterBranchBSB → BranchAddress

Since BranchAddress is transitively dependent on the primary key, EmpNo, we have to move it to a new table.

New relations
Employee(<u>EmpNo</u>, RegisterBranchBSB*)
Branch (<u>RegisterBranchBSB</u>, BranchAddress)

All new relations above are now decomposed to 3NF as they are fully dependent on the whole PK in each case and not transitively dependent on the primary key.

Q4

Final schemas:

Compiling the new relations that were created above:

Customer (CustNo, CustName, Phone, Address, DateOfBirth, EmailAddress)
Account (AccNo, CustNo*, EmpNo*)
~~Customer1(CustNo, CustName)~~
Employee(EmpNo, RegisterBranchBSB*)
Branch (RegisterBranchBSB, BranchAddress)
AccType (TypeID, TypeName)
~~Customer2 (CustNo, CustName)~~
~~AccType (TypeID, TypeName)~~
CustomerACCType(CustNo*, TypeID*)

The AccType relation is a duplicate and can be removed. Customer1 and Customer 2 relations have the same Pk as Customer relation can therefore can be combined as a single relation.

The final schema is as follows:

Customer (CustNo, CustName, Phone, Address, DateOfBirth, EmailAddress)
Account (AccNo, CustNo*, EmpNo*)
Employee(EmpNo, RegisterBranchBSB*)
Branch (RegisterBranchBSB, BranchAddress)
AccType (TypeID, TypeName)
CustomerACCType(CustNo*, TypeID*)

PART B

/*Q1 */

```
SELECT title AS 'Title', subtitle AS 'Subtitle', Edition,place AS 'Place'

FROM book

WHERE place = 'OXFORD';
```

/*Q2 */

```
SELECT bookdescid,count(*) AS 'Number of book copies'

FROM book_copy

GROUP BY bookdescid

ORDER BY count(*) DESC;
```

/*Q2 - BONUS*/

```
SELECT book.BOOKDESCID, count(book_copy.BOOKID) AS 'Number of book copies'

FROM book

        LEFT JOIN book_copy

                ON book.BOOKDESCID = book_copy.BOOKID

GROUP BY book.BOOKDESCID

HAVING count(book_copy.BOOKID) = 0;
```

/* Q3 a.*/

```
SELECT DISTINCT b.bookdescid AS 'Book Desc ID',b.title AS 'Title',b.subtitle AS 'Subtitle',b.edition AS

         'Edition',b.place AS 'Place'

FROM book b

        LEFT JOIN (SELECT bookdescid,bc.bookid,bcc.bookid

                    FROM book_copy bc

                        JOIN borrow_copy bcc
```

```
                                    ON bc.bookid = bcc.bookid)AS bb
                            ON b.bookdescid = bb.bookdescid
WHERE bb.bookdescid IS NULL;
```

/*Q3 b. */

```
SELECT DISTINCT b.bookdescID, b.title AS 'Title', b.subtitle AS 'Subtitle', b.edition AS 'Edition',
                                b.place AS 'Place'
FROM book b
WHERE b.bookdescID NOT IN (
        SELECT DISTINCT bc.bookdescID
        FROM book_copy bc
        WHERE bc.bookID IN (
                SELECT DISTINCT bcc.bookID
                FROM borrow_copy bcc));
```

/*Q4*/

```
SELECT authorID, firstname, middlename, lastname, count(*) AS 'No. of books'
FROM written_by
  JOIN author
    ON written_by.AUTHORID=author.AUTHORID
WHERE lower(ROLE) = 'author'
GROUP BY written_by.AUTHORID
HAVING count(*) >1;
```

/*Q5*/

```
SELECT book.TITLE AS 'Book Title',date(borrow.DUEDATE) AS 'Due Date',date(borrow.RETURNDATE)
AS 'Return Date',
  (borrow.RETURNDATE-borrow.DUEDATE) AS 'No of days delayed'
FROM borrow
```

```
    JOIN borrow_copy
        ON borrow.TRANSACTIONID=borrow_copy.TRANSACTIONID
            JOIN book_copy
                ON book_copy.BOOKID= borrow_copy.BOOKID
                    JOIN book
                        ON book_copy.BOOKDESCID=book.BOOKDESCID
WHERE borrow.returndate > borrow.DUEDATE;
```

/* Q6 */

```
SELECT p.PUBLISHERFULLNAME
FROM publisher p
WHERE p.PUBLISHERID NOT IN
              (SELECT pb.PUBLISHERID
              FROM published_by pb
              WHERE lower(role) = 'editor')
ORDER BY publisherfullname;
```

/*Q7*/

```
SELECT a.firstname || ' ' || a.lastname AS ' Author', b.TITLE AS 'Title',
    s.SUBJECTTYPE AS 'Subject',b.YEAR AS 'Published Year'
FROM book b
    LEFT JOIN written_by w
        ON b.BOOKDESCID= w.BOOKDESCID
            LEFT JOIN author a
                ON a.AUTHORID= w.AUTHORID
                    LEFT JOIN subject s
                        ON b.SUBJECTID=s.SUBJECTID
```

```
WHERE lower(w.ROLE) = 'author' AND
   lower(s.SUBJECTTYPE) = 'artificial intelligence';
```

/*Q8*/

```
SELECT b.TITLE AS 'Title',p.FIRSTNAME || ' ' || p.LASTNAME AS ' Borrower Name',
   date(bo.borrowdate) AS 'Date of Borrow'
FROM book b
   LEFT JOIN book_copy bc
      ON b.BOOKDESCID = bc.BOOKDESCID
         LEFT JOIN borrow_copy c
            ON bc.BOOKID = c.BOOKID
               LEFT JOIN borrow bo
                  ON c.TRANSACTIONID= bo.TRANSACTIONID
                     LEFT JOIN person p
                        ON bo.PERSONID = p.PERSONID
WHERE b.TITLE != 'COMPUTER SCIENCE' AND
    p.PERSONID IN
       (SELECT p.personid
       FROM book b
          LEFT JOIN book_copy bc
             ON b.BOOKDESCID = bc.BOOKDESCID
               LEFT JOIN borrow_copy c
                  ON bc.BOOKID = c.BOOKID
                     LEFT JOIN borrow bo
                        ON c.TRANSACTIONID= bo.TRANSACTIONID
                           LEFT JOIN person p
                              ON bo.PERSONID = p.PERSONID
       WHERE b.TITLE = 'COMPUTER SCIENCE');
```

/*Q8 BONUS*/

```
SELECT b.bookdescID, b.title AS 'Title', b.year AS 'Year of publication',

        COUNT(b.bookdescID) AS 'No of times borrowed'

FROM book b

        JOIN book_copy bc

                ON b.bookdescID = bc.bookdescID

                        JOIN borrow_copy bcc

                                ON bc.bookID = bcc.bookID

GROUP BY b.bookdescID

HAVING COUNT(b.bookdescID) > 0

ORDER BY COUNT(b.bookdescID) DESC;
```