Database Management Systems Lab

CSE 4308

NAME: NAZIA KARIM KHAN OISHEE

ID : 200042137

Introduction

In this lab our task was to implement multiple SQL statements and execute them to get a grip over data definition and data manipulation.

Solution of the task

Our first task was to write SQL statements to create the tables with the given specifications.

So I created the tables using the 'CREATE TABLE' command. Here in ACCOUNT table ACCOUNT_NO was the primary key, in CUSTOMER table CUSTOMER_NO was the primary key and in DEPOSITOR table ACCOUNT_NO ,CUSTOMER_NO both were the primary key.

```
CREATE TABLE ACCOUNT(
    ACCOUNT_NO varchar(5),
    BALANCE Number NOT NULL,
    CONSTRAINT PK_Account_No PRIMARY KEY ( ACCOUNT_NO )
) ;
CREATE TABLE CUSTOMER_(
    CUSTOMER_NO varchar(5),
    CUSTOMER_NAME varchar(20) NOT NULL,
    CUSTOMER_CITY varchar(10),
    CONSTRAINT PK_Customer_No PRIMARY KEY (CUSTOMER_NO)
);
CREATE TABLE DEPOSITOR(
ACCOUNT_NO varchar(5),
```

```
CUSTOMER_NO varchar(5),

CONSTRAINT PK_DEPOSITOR PRIMARY KEY (ACCOUNT_NO , CUSTOMER_NO)

);
```

'CREATE TABLE' command falls under the categorization of DDL or data definition language as we are defining a table, its attributes, and structure through it.

After that we had to write SQL statements to perform multiple alteration operations:

(a) Given task: Add a new attribute 'DATE_OF_BIRTH' in the CUSTOMER table.

Solution:

```
ALTER TABLE CUSTOMER_ ADD DATE_OF_BIRTH DATE;
```

(b) Given task: Modify the data type of BALANCE from NUMBER to NUMBER(12, 2).

Solution:

```
ALTER TABLE ACCOUNT MODIFY BALANCE NUMBER(12,2);
```

(c) Given task: Rename the attribute ACCOUNT_NO, CUSTOMER_NO from the DEPOSITOR table to A_NO and C_NO, respectively .

Solution:

```
ALTER TABLE DEPOSITOR RENAME COLUMN ACCOUNT_NO TO A_NO;

ALTER TABLE DEPOSITOR RENAME COLUMN CUSTOMER_NO TO C_NO;
```

(d)Given task: Rename the table DEPOSITOR to DEPOSITOR_INFO.

Solution:

```
ALTER TABLE DEPOSITOR RENAME TO DEPOSITOR_INFO;
```

(e)Given task: Add two foreign key constraints FK_DEPOSITOR_ACCOUNT and FK_DEPOSITOR_CUSTOMER that identifies A_NO and C_NO as foreign keys.

Solution:

```
ALTER TABLE DEPOSITOR_INFO ADD CONSTRAINT FK_DEPOSITOR_ACCOUNT_NO FOREIGN KEY
(A_NO) REFERENCES ACCOUNT(ACCOUNT_NO);

ALTER TABLE DEPOSITOR_INFO ADD CONSTRAINT FK_DEPOSITOR_CUSTOMER_NO FOREIGN KEY
(C_NO) REFERENCES CUSTOMER_(CUSTOMER_NO);
```

Through the foreign keys relationship is established among DEPOSITOR_INFO, ACCOUNT and CUSTOMER. Foreign keys play an important role in building relational database model.

These altering commands also fall under the categorization of DDL.

Then I inserted values in these tables.

```
INSERT INTO ACCOUNT VALUES('A_01',5000);

INSERT INTO ACCOUNT VALUES('A_02',10000);

INSERT INTO ACCOUNT VALUES('A_03',50000);

INSERT INTO CUSTOMER_ VALUES('C_01','NAZIA','KHL','1-JAN-2000');

INSERT INTO CUSTOMER_ VALUES('C_02' , 'KHAN', 'DHK','14-JAN-2002');

INSERT INTO DEPOSITOR_INFO VALUES ('A_01','C_01');

INSERT INTO DEPOSITOR_INFO VALUES ('A_02','C_02');
```

Then I performed the given queries using SQL statements through DML:

(a) Given query:Find all account numbers with balances less than 100000.

Solution:

```
SELECT ACCOUNT_NO FROM ACCOUNT WHERE BALANCE < 100000;
```

(b) Given query: Find all customer names who live in ‘KHL’ city.

Solution:

```
SELECT CUSTOMER_NAME FROM CUSTOMER_ WHERE CUSTOMER_CITY = 'KHL';
```

(c) Given query: Find all customer numbers whose name contains ‘A’.

Solution:

```
SELECT CUSTOMER_NO FROM CUSTOMER_ WHERE CUSTOMER_NAME LIKE '%A%';
```

(d)Given query: Find distinct account numbers from the DEPOSITOR_INFO table.

Solution:

```
SELECT DISTINCT A_NO FROM DEPOSITOR_INFO;
```

(e)Given query: Show the result of Cartesian Product between ACCOUNT and DEPOSITOR_INFO table.

Solution:

```
SELECT * FROM ACCOUNT, DEPOSITOR_INFO;
```

(f)Given query: Show the result of Natural Join between CUSTOMER and DEPOSITOR_INFO table.

Solution:

```
SELECT * FROM CUSTOMER_,DEPOSITOR_INFO WHERE CUSTOMER_NO=C_NO;
```

(g)Given query: Find all customer names and their city who have an account.

Solution:

```
SELECT CUSTOMER_.CUSTOMER_NAME, CUSTOMER_.CUSTOMER_CITY FROM CUSTOMER_,
DEPOSITOR_INFO WHERE CUSTOMER_NO=C_NO;
```

(h) Given query: Find all customer related information who have balances greater than 1000.

Solution:

```
SELECT DISTINCT CUSTOMER_.CUSTOMER_NAME, CUSTOMER_.CUSTOMER_NO,
CUSTOMER_.CUSTOMER_CITY,CUSTOMER_.DATE_OF_BIRTH FROM
CUSTOMER_,DEPOSITOR_INFO,ACCOUNT WHERE (CUSTOMER_NO=C_NO AND ACCOUNT_NO=A_NO AND
BALANCE>1000);
```

(i) Given query: Find all accounts related information where the balance is between 5000 and 10000 or their depositor lives in 'DHK' city.

Solution:

```
SELECT DISTINCT ACCOUNT.ACCOUNT_NO,ACCOUNT.BALANCE FROM
ACCOUNT,DEPOSITOR_INFO,CUSTOMER_ WHERE (BALANCE>=5000 AND BALANCE<=10000) OR
(ACCOUNT_NO = A_NO AND C_NO=CUSTOMER_NO AND CUSTOMER_CITY='DHK');
```

Conclusion

While implementing SQL statements it is more important to get the logic underneath the working code rather than syntax. Syntax of SQL is human friendly enough to understand. So understanding the logic behind how to get the queries by accessing multiple tables, when to access which table under which condition was my core concern while completing the task. It took me a while to get that but eventually I could do it successfully.