Islamic University of Technology (IUT)

Report on Lab 03

Submitted By

Shanta Maria (ID:200042172)

CSE 4308 Database Management Systems Lab

Submitted To

Md. Bakhtiar Hasan

Lecturer, Department of CSE

Zannatun Naim Srsity

Lecturer, Department of CSE

September 7, 2022

Introduction

In the lab class, we were given three tasks to solve using SQL command line to understand the basics of data definition and data manipulation.

1 Task 1

Write SQL statements to create the following tables with the given specifications:

(a) ACCOUNT

ACCOUNT_NO	CHAR(5)	(e.g.: A-101) Primary Key
BALANCE	NUMBER	Not Null

(b) CUSTOMER

CUSTOMER_NO	CHAR(5)	(e.g.: C-101) Primary Key
CUSTOMER_NAME	VARCHAR2(20)	Not Null
CUSTOMER_CITY	VARCHAR2(20)	(e.g.: DHK, KHL, etc.)

(c) DEPOSITOR

ACCOUNT_NO	CHAR(5)	(e.g.: A-101)
CUSTOMER_NO	CHAR(5)	(e.g.: C-101)
		Primary Key(ACCOUNT_NO, CUSTOMER_NO)

1.1 Solution

```
CREATE TABLE ACCOUNT
(
  ACCOUNT_NO CHAR(5),
  BALANCE NUMBER NOT NULL,
  CONSTRAINT PK_ACCOUNT_NO PRIMARY KEY(ACCOUNT_NO)
);

CREATE TABLE CUSTOMER
(
    CUSTOMER_NO CHAR(5),
```

```
    CUSTOMER_NAME VARCHAR(20) NOT NULL,
    CUSTOMER_CITY VARCHAR(10),
    CONSTRAINT PK_CUSTOMER_NO PRIMARY KEY(CUSTOMER_NO)
);

CREATE TABLE DEPOSITOR
(
    ACCOUNT_NO CHAR(5),
    CUSTOMER_NO CHAR(5),
    CONSTRAINT PK_DEPOSITOR_NO PRIMARY KEY(ACCOUNT_NO, CUSTOMER_NO)
);
```

1.2 Analysis and Explanation

This task was easy to complete since we already learned how to create tables in the last lab. Assigning primary keys was a new concept but I was able to implement it.

1.3 Difficulties

I did not face any difficulties when doing this task and no mentionable issues were encountered.

```
SQL> CREATE TABLE ACCOUNT(
  2    ACCOUNT_NO CHAR(5),
  3    BALANCE NUMBER NOT NULL,
  4    CONSTRAINT PK_ACCOUNT_NUM PRIMARY KEY(ACCOUNT_NO)
  5   );

Table created.

SQL>
SQL> CREATE TABLE CUSTOMER(
  2      CUSTOMER_NO CHAR(5),
  3      CUSTOMER_NAME VARCHAR(20) NOT NULL,
  4      CUSTOMER_CITY VARCHAR(10),
  5      CONSTRAINT PK_CUSTOMER_NO PRIMARY KEY(CUSTOMER_NO)
  6  );

Table created.

SQL>
SQL> CREATE TABLE DEPOSITOR(
  2      ACCOUNT_NO CHAR(5),
  3      CUSTOMER_NO CHAR(5),
  4      CONSTRAINT PK_DEPOSITOR_NO PRIMARY KEY(ACCOUNT_NO, CUSTOMER_NO)
  5  );

Table created.
```

Figure 1: Task 1

2 Task 2

Write SQL statements to perform the following alteration operations:

(a) Add a new attribute 'DATE_OF_BIRTH' (DATE type) in CUSTOMER table.

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

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

(d) Rename the table DEPOSITOR to DEPOSITOR_INFO.

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

2.1 Solution

```
ALTER TABLE CUSTOMER ADD DATE_OF_BIRTH DATE;


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

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

ALTER TABLE DEPOSITOR RENAME COLUMN CUSTOMER_NO TO C_NO;

ALTER TABLE DEPOSITOR RENAME TO DEPOSITOR_INFO;

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

ALTER TABLE DEPOSITOR_INFO ADD CONSTRAINT FK_DEPOSITOR_CUSTOMER FOREIGN
KEY(C_NO) REFERENCES CUSTOMER(CUSTOMER_NO);
```

2.2 Analysis and Explanation

This task was completed easily by following the instructions in the PDF document.

2.3 Difficulties

I did face some difficulties when doing task (e). I got an error 3 times when trying to implement this task because I inputted the referenced table key as C_NO instead of CUSTOMER_NO. This part was a bit confusing and hard to understand for me.

```
SQL> ALTER TABLE CUSTOMER ADD DATE_OF_BIRTH DATE;

Table altered.

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

Table altered.

SQL>
SQL> ALTER TABLE DEPOSITOR RENAME COLUMN ACCOUNT_NO TO A_NO;

Table altered.

SQL>
SQL> ALTER TABLE DEPOSITOR RENAME COLUMN CUSTOMER_NO TO C_NO;

Table altered.

SQL>
SQL> ALTER TABLE DEPOSITOR RENAME TO DEPOSITOR_INFO;

Table altered.
```

Figure 2: Task 2-1

```
SQL>
SQL> ALTER TABLE DEPOSITOR_INFO ADD CONSTRAINT FK_DEPOSITOR_ACCOUNT FOREIGN KEY(A_NO) REFERENCES ACCOUNT(ACCOUNT_NO);

Table altered.

SQL>
SQL> ALTER TABLE DEPOSITOR_INFO ADD CONSTRAINT FK_DEPOSITOR_CUSTOMER FOREIGN KEY(C_NO) REFERENCES CUSTOMER(CUSTOMER_NO);
```

Figure 3: Task 2-2

3 Task 3

Write SQL statements to answer the following queries:

(a) Find all account number with balance greater than 100000.

(b) Find all customer names who live in ‘Dhaka’ city.

(c) Find all customer number whose name starts with ‘A’.

(d) Find distinct account numbers from DEPOSITOR_INFO table.

(e) Show the result of cartesian product between ACCOUNT and DEPOSITOR_INFO table.

(f) Show the result of natural join between CUSTOMER and DEPOSITOR_INFO table.

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

(h) Find all customer related information who have balance greater than 1000.

(i) Find all accounts related information where balance is in between 5000 and 10000 and their depositor lives in ‘Khulna’ city.

3.1 Solution

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

SELECT CUSTOMER_NAME
 FROM CUSTOMER
 WHERE CUSTOMER_CITY = 'DHK';

SELECT CUSTOMER_NO
 FROM CUSTOMER
 WHERE CUSTOMER_NAME LIKE 'A%';

SELECT DISTINCT A_NO
 FROM DEPOSITOR_INFO;

SELECT *
 FROM ACCOUNT, DEPOSITOR_INFO;

SELECT *
 FROM CUSTOMER
 NATURAL JOIN DEPOSITOR_INFO;

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

SELECT CUSTOMER.CUSTOMER_NO, CUSTOMER.CUSTOMER_NAME, CUSTOMER.CUSTOMER_CITY,
```

```
CUSTOMER.DATE_OF_BIRTH
 FROM CUSTOMER, DEPOSITOR_INFO, ACCOUNT
 WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR_INFO.C_NO AND ACCOUNT.BALANCE>1000
 AND ACCOUNT.ACCOUNT_NO = DEPOSITOR_INFO.A_NO;
```

```
SELECT ACCOUNT.ACCOUNT_NO, ACCOUNT.BALANCE
 FROM CUSTOMER, DEPOSITOR_INFO, ACCOUNT
 WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR_INFO.C_NO AND ACCOUNT.BALANCE<10000
 AND ACCOUNT.BALANCE>5000 AND ACCOUNT.ACCOUNT_NO = DEPOSITOR_INFO.A_NO AND
 CUSTOMER.CUSTOMER_CITY = 'KHL';
```

3.2 Analysis and Explanation

Task(a) was easy to implement.

Task(b) was tricky since `CUSTOMER_COUNTRY` was given as 'Dhaka' but we had to input it as 'DHK'.

Task(c) was also easy to implement since it was taught during theory DBMS classes. Here `'A%'` means `A` would be at the start of the string followed by any string.

Task(d),(e) and (f) was solvable. The use of `'distinct'` keyword in task (d) was learned in the last lab class. The concept and code for cartesian product and natural join was taught in the theory as well as the lab classes.

In task (g), (h) and (i), we had to manipulate information from 3 tables. For (g), we checked if the `CUSTOMER_NO` in the `CUSTOMER` table matched any `C_NO` in `DEPOSITOR_INFO` table to confirm the existence of a customer with an account. Similarly, we also checked if the account in `ACCOUNT` table matches the one in `DEPOSITOR_INFO` and `CUSTOMER` table so that we can access the `CUSTOMER_CITY` attribute to check if the person is from 'Khulna' city which is checked as

'KHL' and the BALANCE attribute to check if their balance is between 5000 and 10000.

3.3 Difficulties

I faced difficulties when implementing task (g), (h) and (i). Accessing information and relating those information from three different tables were difficult and confusing. I did not understand the use of DEPOSITOR_INFO table to check account existence at first. Later, I understood that the table was used to verify if the account actually existed otherwise there would be no account information for that account in the ACCOUNT table.

```
SQL> SELECT ACCOUNT_NO
  2    FROM ACCOUNT
  3   WHERE BALANCE>100000;

no rows selected

SQL>
SQL>  SELECT CUSTOMER_NAME
  2    FROM CUSTOMER
  3   WHERE CUSTOMER_CITY = 'DHK';

no rows selected

SQL>
SQL>  SELECT CUSTOMER_NO
  2    FROM CUSTOMER
  3   WHERE CUSTOMER_NAME LIKE 'A%';

no rows selected

SQL>
SQL>  SELECT DISTINCT A_NO
  2    FROM DEPOSITOR_INFO;

no rows selected
```

Figure 4: Task 3-1

```
SQL>  SELECT *
  2    FROM ACCOUNT, DEPOSITOR_INFO;

no rows selected

SQL>
SQL>  SELECT *
  2    FROM CUSTOMER
  3    NATURAL JOIN DEPOSITOR_INFO;

no rows selected

SQL>
SQL>  SELECT CUSTOMER_NAME, CUSTOMER_CITY
  2    FROM CUSTOMER, DEPOSITOR_INFO
  3    WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR_INFO.C_NO;

no rows selected
```

Figure 5: Task 3-2

```
SQL>
SQL>  SELECT CUSTOMER.CUSTOMER_NO, CUSTOMER.CUSTOMER_NAME, CUSTOMER.CUSTOMER_CITY
  2    FROM CUSTOMER, DEPOSITOR_INFO, ACCOUNT
  3    WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR_INFO.C_NO AND ACCOUNT.BALANCE>1000 AND ACCOUNT.ACCOUNT_NO = DEPOSITOR_INFO
.A_NO;

no rows selected

SQL>
SQL>
SQL>  SELECT ACCOUNT.ACCOUNT_NO, ACCOUNT.BALANCE
  2    FROM CUSTOMER, DEPOSITOR_INFO, ACCOUNT
  3    WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR_INFO.C_NO AND ACCOUNT.BALANCE<100000 AND ACCOUNT.BALANCE>5000 AND ACCOUNT.
ACCOUNT_NO = DEPOSITOR_INFO.A_NO AND CUSTOMER.CUSTOMER_CITY = 'KHL';

no rows selected
```

Figure 6: Task 3-3

Conclusion

As shown in the report, I have solved and tested the solutions for all three tasks given in the lab. All the commands used were written in notepad which was then saved with .sql extension. The .sql file was then run through the SQL command line to execute all the commands.