Islamic University of Technology

Lab 03

CSE 4308 - DBMS Lab

<u>Submitted To :</u>

Md. Bakhtiar Hasan
Lecturer, CSE Department
Islamic University of Technology

<u>Submitted By :</u>

Rhidwan Rashid
ID : 200042149
Prog. : SWE
Dept. : CSE

Creating tables:

Working codes:
For ACCOUNT table:

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

For CUSTOMER table:

```
CREATE TABLE CUSTOMER(
     CUSTOMER_NO CHAR(5),
     CUSTOMER_NAME VARCHAR2(20) NOT NULL,
     CUSTOMER_CITY VARCHAR2(10),
     CONSTRAINT PK_CUSTOMER_NO PRIMARY KEY(CUSTOMER_NO)
);
```

For DEPOSITOR table:

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

Analysis: The problem requires creating table with primary key and not null attributes.

Explanation: First I created the tables with simple syntax, then added additional constraints. For ACCOUNT table, I added constraint primary key and made ACCOUNT_NO a primary key. BALANCE field can't be null, so, NOT NULL must be added at the end. For CUSTOMER table the CUSTOMER_NO attribute was made a primary key and CUSTOMER_NAME was set to not null. For DEPOSITOR table, both CUSTOMER_NO and ACCOUNT_NO attribute was made primary key.

Alternation Operations:

(a) Analysis: The problem requires adding a new attribute date of birth in CUSTOMER table.

Working Code:

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

Explanation : First I used ALTER TABLE keyword, then table name, then ADD keyword and lastly attribute name(DATE_OF_BIRTH) and attribute type(DATE).

(b) Analysis: The problem requires modifying an data type of a table.

Working Code:

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

Explanation: The ALTER TABLE keyword was used as Before, then table name, then MODIFY keyword and after that attribute name(BALANCE) and the new type we want to assign.

(c) Analysis: The problem require renaming attributes of a table.

Working code:

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

Explanation: The ALTER TABLE keyword is same in every alternation operation. The keyword for renaming is RENAME COLUMN, then the column name we want to rename, the TO keyword and after that the new name for the column.

(d) Analysis: The problem requires renaming a table.

Working code:

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

Explanation: Table name after ALTER TABLE keyword then RENAME TO keyword for renaming the table and after that we write the new name for the table.

(e) Analysis: The problem requires adding foreign keys as constraints.

Working code:

```
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);
```

Explanation: After writing alter table and table name I used ADD CONSTRAINT keyword for adding new constraints, then I named the constrained and wrote A_NO attribute inside FOREIGN KEY keyword to set it as a foreign key, then I added reference to ACCOUNT_NO which is an attribute of ACCOUNT table. The same process for second operation.

Findings: Reference need to be added to make an attribute foreign key.

Queries:

(a) Analysis: The problem requires finding all account number with balance less than 100000.

Working code:
```
SELECT ACCOUNT_NO
FROM ACCOUNT
WHERE BALANCE<100000;
```

Explanation: I selected ACCOUNT_NO attribute with SELECT keyword from ACCOUNT table with FROM keyword and added the condition with WHERE keyword.

(b) Analysis: The problem requires finding all customer names who live in "KHL" city.

Working code:
```
SELECT CUSTOMER_NAME
FROM CUSTOMER
WHERE CUSTOMER_CITY = 'KHL';
```

Explanation: I selected CUSTOMER_NAME attribute with SELECT keyword from CUSTOMER table with FROM keyword and added the condition "CUSTOMER_CITY = 'KHL' with WHERE keyword.

(c) Analysis: The problem requires finding all customer number whose name contain 'A'.

Working code:

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

Explanation: I selected CUSTOMER_NO attribute with SELECT keyword from CUSTOMER table with FROM keyword and added the condition "CUSTOMER_NAME LIKE '%A%' with WHERE keyword. LIKE keyword is used for finding names which contains 'A'.

(d) Analysis: The problem requires finding distinct account numbers from DEPOSITOR_INFO table.

Working code:

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

Explanation: I selected A_NO attribute(which was renamed in alternation operation) with SELECT keyword from DEPOSITOR_INFO table with FROM keyword. DISTINCT keyword was used to select only distinct numbers.

(e) Analysis: The problem requires showing the result of cartesian product between ACCOUNT and DEPOSITOR_INFO table.

Working code:
```
SELECT *
FROM ACCOUNT, DEPOSITOR_INFO;
```

Explanation: I selected all attributes with SELECT keyword and '*' from ACCOUNT table and DEPOSITOR_INFO table with FROM keyword. Selecting from two tables shows the result of cartesian product.

(f) Analysis: The problem requires showing the result of natural join between CUSTOMER and DEPOSITOR_INFO table

Working code:
```
SELECT *
FROM CUSTOMER
NATURAL JOIN DEPOSITOR_INFO;
```

Explanation: I selected all attributes with SELECT keyword and '*' from CUSTOMER table with FROM keyword and joined the DEPOSITOR_INFO table with NATURAL JOIN keyword.

(g) Analysis: The problem requires finding all customer names who have an account and the city they live in

Working code:
```
SELECT CUSTOMER_NAME, CUSTOMER_CITY
FROM CUSTOMER, ACCOUNT, DEPOSITOR_INFO
WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR.C_NO AND
ACCOUNT.ACCOUNT_NO = DEPOSITOR.A_NO;
```

Explanation: I selected CUSTOMER_NAME and CUSTOMER_CITY attribute with SELECT keyword from CUSTOMER, ACCOUNT and DEPOSITOR_INFO table with FROM keyword(this cross joins all the table). Then we filter out the values with "`WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR.C_NO AND ACCOUNT.ACCOUNT_NO = DEPOSITOR.A_NO`" statement.

(h) Analysis: The problem requires finding all customer related information who have balance greater than 1000.

Working code:

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

Explanation: I selected CUSTOMER_NO, CUSTOMER_NAME and CUSTOMER_CITY attribute with SELECT keyword from CUSTOMER, ACCOUNT and DEPOSITOR_INFO table with FROM keyword(this cross joins all the table). Then we filter out the values with "`WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR.C_NO AND ACCOUNT.ACCOUNT_NO = DEPOSITOR.A_NO AND ACCOUNT.BALANCE > 1000`" statement.

(h) Analysis: The problem requires finding all account related information where balance is in between 5000 and 10000 or their depositor lives in 'DHK' city.

Working code:

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

Explanation: I selected ACCOUNT_NO and BALANCE attribute with SELECT keyword from CUSTOMER, ACCOUNT and DEPOSITOR_INFO table with FROM keyword(this cross joins all the table). Then we filter out the values with "`WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR.C_NO AND ACCOUNT.ACCOUNT_NO = DEPOSITOR.A_NO AND ACCOUNT.BALANCE>=5000 AND ACCOUNT.BALANCE<=10000 OR CUSTOMER.CUSTOMER_CITY='DHK'`" `statement.`

Overall findings and problems: All the queries were easy except the last three. I had to add many conditions in where statement. I found out that there is a precedence issue in conditions from (h) problem. The issue was solved after writing AND conditions first, then writing the OR condition.