Islamic University of Technology

Lab Task 03

CSE 4308 - DBMS Lab

Submitted To :

MD Bakhtiar Hasan
Lecturer, CSE Department
Islamic University of Technology

Submitted By :

Sian Ashsad
ID : 200042151
Prog. : SWE
Dept. : CSE

Task 1 :

Working code :

```
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 VARCHAR2 (20) NOT NULL,
    CUSTOMER_CITY VARCHAR2 (10),
    CONSTRAINT PK_CUSTOMER_NO  PRIMARY KEY (CUSTOMER_NO)
);

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

Report :

Analysis : Here the key objective is to create 3 tables with given attributes.

Explanation of solution : First I will create the Account table with given attributes. I will add CONSTRAINT on the PRIMARY KEY (ACCOUNT_NO). This will ensure the uniqueness of the primary key. In the same way I will create the remaining tables. The DEPOSITOR table will have 2 primary keys.

Findings : CONSTRAINT command was new to me. It’s implementation process was also something I found out while doing the task.

Problems : There were no issues in this task.

Task 2 :

Working code :

```sql
--A
ALTER TABLE CUSTOMER ADD DATE_OF_BIRTH DATE;
--B
ALTER TABLE ACCOUNT MODIFY BALANCE NUMBER(12,2);
--C
ALTER TABLE DEPOSITOR RENAME COLUMN ACCOUNT_NO TO A_NO;
ALTER TABLE DEPOSITOR RENAME COLUMN CUSTOMER_NO TO C_NO;
--D
ALTER TABLE DEPOSITOR RENAME TO DEPOSITOR_INFO;
--E
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);
```

Report :

Analysis : Here the key objective is to modify the tables according to the given alternations.

Explanation of solution :

A) Adds a DATE_OF_BIRTH column to the CUSTOMER table of DATE data type.
B) Modifies the BALANCE attribute of table ACCOUNT by changing its datatype to NUMBER(12,2)
C) Changes of the names of the columns ACCOUNT_NO and CUSTOMER_NO to A_NO and C_NO respectively.
D) Changes the name of the table DEPOSITOR to DEPOSITOR_INFO.
E) Here A_NO and C_NO have been assigned as foreign keys which references ACCOUNT_NO of ACCOUNT table and CUSTOMER_NO of CUSTOMER respectively.

Findings : Assigning foreign keys and adding references to appropriate columns must be done with caution. To modify the data type of an attribute, that attribute must have no values stored inside of it. The naming convention of CONSTRAINT must be followed : PK_COLUMN/FK_COLUMN.

Problems : By following the instructions properly, there were no issues completing the tasks.

Task 3 :

Working code :

```sql
INSERT INTO ACCOUNT VALUES ('A-101', 69420);
INSERT INTO ACCOUNT VALUES ('A-102', 19900);
INSERT INTO ACCOUNT VALUES ('A-103', 9990);
INSERT INTO ACCOUNT VALUES ('A-104', 123445678);
INSERT INTO ACCOUNT VALUES ('A-105', 1);
INSERT INTO ACCOUNT VALUES ('A-106', 42069);
INSERT INTO ACCOUNT VALUES ('A-107', 404404);

INSERT INTO CUSTOMER VALUES ('C-101', 'SIAN', 'DHK', '31-JUL-01');
INSERT INTO CUSTOMER VALUES ('C-102', 'DIHAN', 'KHL', '07-AUG-01');
INSERT INTO CUSTOMER VALUES ('C-103', 'AKASH', 'DHK', '29-APR-01');
INSERT INTO CUSTOMER VALUES ('C-104', 'ANAN', 'KHL', '19-JUL-01');
INSERT INTO CUSTOMER VALUES ('C-105', 'NOWRID', 'KHL', '1-APR-00');
INSERT INTO CUSTOMER VALUES ('C-106', 'ONGKON', 'DHK', '31-JUL-01');

INSERT INTO DEPOSITOR_INFO VALUES ('A-101', 'C-101');
INSERT INTO DEPOSITOR_INFO VALUES ('A-102', 'C-102');
INSERT INTO DEPOSITOR_INFO VALUES ('A-103', 'C-103');
INSERT INTO DEPOSITOR_INFO VALUES ('A-104', 'C-104');
INSERT INTO DEPOSITOR_INFO VALUES ('A-105', 'C-105');
INSERT INTO DEPOSITOR_INFO VALUES ('A-106', 'C-106');

--A
SELECT ACCOUNT_NO FROM ACCOUNT WHERE BALANCE < 100000;
--B
SELECT CUSTOMER_NAME FROM CUSTOMER WHERE CUSTOMER_CITY = 'KHL';
--C
SELECT CUSTOMER_NO FROM CUSTOMER WHERE CUSTOMER_NAME LIKE '%A%';
--D
SELECT DISTINCT A_NO FROM DEPOSITOR_INFO ORDER BY A_NO ASC;
--E
SELECT * FROM ACCOUNT, DEPOSITOR_INFO;
--F
```

```sql
SELECT * FROM CUSTOMER NATURAL JOIN DEPOSITOR_INFO;
--G
SELECT CUSTOMER_NAME, CUSTOMER_CITY
FROM DEPOSITOR_INFO,CUSTOMER,ACCOUNT
WHERE DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO AND DEPOSITOR_INFO.A_NO =
ACCOUNT.ACCOUNT_NO;
--H
SELECT CUSTOMER_NO,CUSTOMER_NAME, CUSTOMER_CITY
FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO
WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO =
CUSTOMER.CUSTOMER_NO AND ACCOUNT.BALANCE > 1000;
--I
SELECT DISTINCT ACCOUNT_NO,BALANCE
FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO
WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO =
CUSTOMER.CUSTOMER_NO AND CUSTOMER.CUSTOMER_CITY = 'DHK'
OR ACCOUNT.BALANCE BETWEEN 5000 AND 10000;
```

Report :

Analysis : In this task, given records have to complete given queries.

Explanation of solution : First I will input some values into all of the tables.

A) Shows the number of accounts with balances less than 100000.
B) Shows the names of customers who live in KHL.
C) Shows the names of customers who have an "A" in their names.
D) Shows distinct account numbers in an ascending order from DEPOSITOR_INFO table.
E) Shows the result of cross product between ACCOUNT and DEPOSITOR_INFO table.
F) Shows the natural join between CUSTOMER and DEPOSITOR_INFO table.
G) Shows the customer name and city of the customer who has an account. The system checks if the customer's distinct number and account number matches any of the entries in the DEPOSITOR_INFO table.
H) Shows all customer information of the customer who has an account with a balance greater than 1000.

I) Shows all account information of the customer who has an account and lives in DHK city or has a balance between 5000 and 10000

Findings : The cross join produces the cross product or Cartesian product of two tables whereas the natural join is based on all the columns having the same name and data types in both the tables. In query (I) the associativity of operators is left associative.

Problems : The only problem faced during this task is on query (I) where I neglected operator associativity. For this, I got a different result than expected. Later on, I followed the proper associativity and fixed the issue.