# Database Management Systems Lab
# Lab 3
# CSE 4308

**A Z Hasnain Kabir**
200042102
Software Engineering

7 September 2022

## Introduction

SQL stands for Structured Query Language. It is used to manage data stored in a Relational Database Management System. It is useful in handling structured data. Structured data incorporates relations among entities and variables. In this lab we are supposed to get started with various Data definition and manipulation techniques.

## Task 1

**Task:** 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(10) | (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) |

**Analysis of the problem:** We need to write some codes in SQL to generate tables according to the given Relation Schema.

## *Code:*

## (a)

## *Code:*

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

## (b)

## *Code:*

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

### (c)

***Code:***

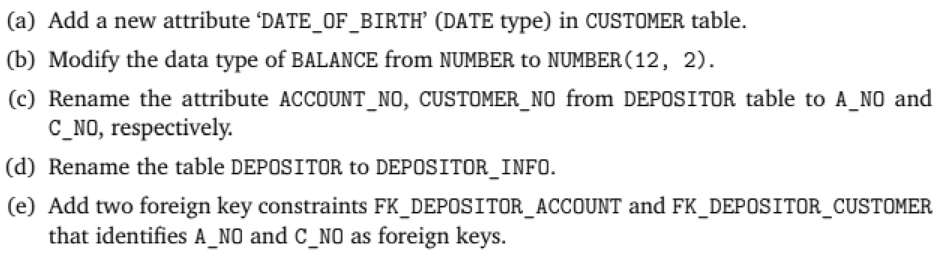
```
CREATE TABLE DEPOSITOR AS(
  SELECT ACCOUNT_NO FROM ACCOUNT,
  SELECT CUSTOMER_NO FROM CUSTOMER,
  CONSTRAINT PK_DEPOSITOR PRIMARY KEY(ACCOUNT_NO,CUSTOMER_NO)
);
```

**Explanation of solution:** CREATE syntax in SQL generates a table according to given schema. ALTER keyword makes changes to relation and we have used it to add primary key to our relation.

**Findings:** We have to log in to our system to make sure the user is created.

## Task 2

**Task:** 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.

**Analysis of the problem:**We have to use ALTER syntax in SQL to make changes to any relation.

***Code:***

### (a)

***Code:***

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

### (b)

***Code:***

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

### (c)

***Code:***

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

### (d)

***Code:***

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

### (e)

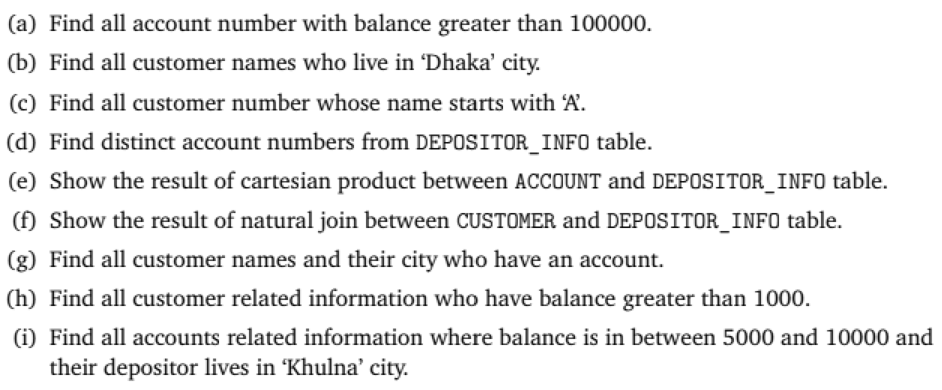
***Code:***

```
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 of solution:** We alter our CUSTOMER relation to add DATE_OF_BIRTH as of type DATE. Similarly we modify BALANCE to precision (12,2). Then we rename DEPOSITOR table columns according to question. We also change the name of DEPOSITOR table to DEPOSITOR_INFO. Lastly, we add two foreign keys to our DEPOSITOR_INFO table called FK_DEPOSITOR_ACCOUNT and FK_DEPOSITOR_CUSTOMER.

## Task 3

**Task:** 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.

**Analysis of the problems:** We are to use SELECT statement multiple times, use cartesian product, where clause etc. to answer given queries.

***Code:***

## (a)

***Code:***

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

## (b)

***Code:***

```
SELECT CUSTOMER_NAME FROM CUSTOMER WHERE CUSTOMER_CITY="Dhaka";
```

## (c)

***Code:***

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

## (d)

***Code:***

```
SELECT DISTINCT(A_NO) FROM DEPOSITOR_INFO;
```

## (e)

***Code:***

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

### (f)

***Code:***

```
SELECT * FROM CUSTOMER NATURAL JOIN DEPOSITOR_INFO;
```

### (g)

***Code:***

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

### (h)

***Code:***

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

### (i)

***Code:***

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

**Explanation of solution:** The solution mainly focuses on selecting rows of data according to our query. In cases from a to c, we select records using where clause. In case d, we use the word distinct to identify unique account numbers. Then, we use cartesian product in order to select reacord from two relations. Also, we use keyword NATURAL JOIN to do a set operation among our relatons. Finally, in cases from g to i, we use cartesian product and select some rows in order to fetch the information we need which is given in the question.