LAB REPORT

Adib Sakhawat

ID: 210042106

BSc. in Software Engineering

Department of Computer Science and Engineering

Islamic University of Technology

CSE 4308

Database Management Systems

LAB 3

31 Aug, 2023

Abstract

This Lab explores the concepts of table Cartesian products and table natural joining. A table Cartesian product is the result of combining two or more tables, and it contains all possible combinations of rows from the tables. A table natural joining is a special type of Cartesian product that only includes rows where the values in the common columns are equal. The assignment begins by introducing the concepts of table Cartesian products and table natural joining. It then provides a step-by-step guide on how to create a table Cartesian product and a table natural join using the SQL language. The assignment also includes a number of exercises that allow students to practice these concepts.

Contents

Chapter 1

Basic Approach to the problem

1.1 Connecting to Database

To approach this problem we have to have oracle installed in our system. From oracle we need to connect to out system user. To do this we have to run the command below.

```
1    CONNECT system/system_password
```

After connecting to system user we need to create a small space for this lab purpose. To do so we need to run the below code sequentially in the oracle command runner.

```
1    CREATE USER lab_3 IDENTIFIED BY lab_3_pass ;
```

Once the this user is created, we need to grant necessary privileges to this user in order to perform the required tasks for this problem. These privileges include creating tables, inserting data, and querying data. Run the following commands to grant all privileges

```
1    GRANT ALL PRIVILEGES TO lab_3;
2    CONNECT lab_3/lab_3_pass;
```

In the second line of the previous code, I have connected my new user to the database.

1.2 Problem Statement

1.2.1 Given Data Table

```
Table Doctor

NAME VARCHAR2(20) (e.g.: MR. X, MR. Y)
SPECIALIZATION CHAR(2) (e.g. CS, GS, IM, ER)
FEE NUMBER (e.g. 2000, 1500)
Primary Key(NAME,
SPECIALIZATION)


Table Patient

PATIENT_NO CHAR(5) (e.g.: P-101) Primary Key
NAME VARCHAR2(20) (e.g.: A, B, C) Not Null
ADDRESS VARCHAR2(10) (e.g.: DHK, KHL, etc.)


Table apoinment

PATIENT_NO CHAR(5) (e.g.: P-101)
NAME VARCHAR2(20) (e.g.: MR. X, MR. Y)
SPECIALIZATION CHAR(2) (e.g. CS, GS, IM, ER)

Primary
Key(PATIENT_NO,
NAME,SPECIALIZATION)
```

1.2.2 Queries on this table to perform

2. Write SQL statements to perform the following alteration operations: (a) Add a new attribute 'APPOINTMENT-DATE' (DATE type) in APPOINTMENT table.

(b) Modify the PRIMARY KEY of APPOINTMENT table and add APPOINTMENT-DATE too with the previous ones.

(c) Rename the attribute PATIENT-NO, NAME from APPOINTMENT table to P-NO and D-NAME respectively.

(d) Rename the table APPOINTMENT to APPOINTMENT-INFO.

(e) Add two foreign key constraints FK-APPOINTMENT-DOCTOR and FK-APPOINTMENT-PATIENT that identifies D-NAME,SPECIALIZATION and P-NO as foreign keys.

3. Insert at least 3 records in each table following the example.

4. Write SQL statements to answer the following queries: (a) Find all the Doctors' names whose fees are less than 1500.

(b) Find all the Patients' names who live in 'KHL' city.

(c) Show the result of Cartesian Product between PATIENT and APPOINTMENT-INFO table.

(d) Show the result of Natural Join between PATIENT and APPOINTMENT-INFO table.

5. Write following DML statements:

(a) Update the NAME and ADDRESS of a tuple from 'A' and 'DHK ' to 'K' and 'RAJ' accordingly.

(b) Update the NAME of table DOCTOR from 'MR. Y' as 'Ms. Y'.

(c) Delete Patient with PATIENT-NO P-101.

(d) Delete all the information without deleting the table structure.

Chapter 2

Table Creation

2.1 Creating Tables and Data Insertion

In this chapter, we explore the process of creating tables and inserting data into them.

2.1.1 Table Creation

We have created three tables: DOCTOR, PATIENT, and APPOINTMENT. Let's examine the creation process for each table.

Creating DOCTOR Table

The following SQL code illustrates the creation of the "DOCTOR" table:

```
1 CREATE TABLE DOCTOR
2 (
3     NAME VARCHAR2(20),
4     SPECIALIZATION VARCHAR2(2),
5     FEE NUMBER,
6     CONSTRAINT DOCTORPK PRIMARY KEY (NAME, SPECIALIZATION)
7 );
```

Listing 2.1: Creating DOCTOR table

The "DOCTOR" table holds details of doctors, such as their names, specializations, and fees. The primary key constraint ("DOCTORPK") ensures each doctor is uniquely identified by their name and specialization.

Creating PATIENT Table

Here's the SQL code for creating the "PATIENT" table:

```
CREATE TABLE PATIENT
(
    PATIENTNO VARCHAR2(5),
    NAME VARCHAR2(20),
    ADDRESS VARCHAR2(10),
    CONSTRAINT PATIENTPK PRIMARY KEY (PATIENTNO)
);
```

Listing 2.2: Creating PATIENT table

The "PATIENT" table stores patient information, including patient numbers, names, and addresses. The primary key constraint ("PATIENTPK") ensures each patient is uniquely identified by their patient number.

Creating APPOINTMENT Table

The following SQL code demonstrates the creation of the "APPOINTMENT" table:

```
CREATE TABLE APPOINTMENT
(
    PATIENTNO VARCHAR2(5),
    NAME VARCHAR2(20),
    SPECIALIZATION VARCHAR2(2),
    CONSTRAINT APPOINTMENTPK PRIMARY KEY (PATIENTNO, NAME,
   SPECIALIZATION)
);
```

Listing 2.3: Creating APPOINTMENT table

The "APPOINTMENT" table manages appointments and includes columns for patient number, doctor's name, and specialization. The primary key constraint ("APPOINTMENTPK") ensures each appointment is uniquely identified by patient number, doctor's name, and specialization.

Chapter 3

Data Insertion

3.0.1 Inserting data into doctor table

```
1 INSERT INTO DOCTOR (NAME, SPECIALIZATION, FEE)
2 VALUES ('MR. X', 'CS', 2000);
```

Listing 3.1: Inserting Data into the DOCTOR Table

Explanation for the first query: This SQL 'INSERT' query adds a new record to the 'DOCTOR' table. The values 'MR. X', 'CS', and 2000 are being inserted into the 'NAME', 'SPECIALIZATION', and 'FEE' columns respectively.

```
1 INSERT INTO DOCTOR (NAME, SPECIALIZATION, FEE)
2 VALUES ('MR. Y', 'GS', 1500);
3 INSERT INTO DOCTOR (NAME, SPECIALIZATION, FEE)
4 VALUES ('MR. Z', 'IM', 1800);
```

Listing 3.2: Inserting More Data into the DOCTOR Table

Explanation for the second and third queries:
Similarly, these 'INSERT' queries insert new records into the 'DOCTOR' table. In the second query, 'MR. Y', 'GS', and 1500 are added, and in the third query, 'MR. Z', 'IM', and 1800 are added.

3.0.2 Inserting data in other table respectively

```
INSERT INTO PATIENT (PATIENT_NO, NAME, ADDRESS)
VALUES ('P-101', 'A', 'DHK');

INSERT INTO PATIENT (PATIENT_NO, NAME, ADDRESS)
VALUES ('P-102', 'B', 'KHL');

INSERT INTO PATIENT (PATIENT_NO, NAME, ADDRESS)
VALUES ('P-103', 'C', 'RAJ');
```

Listing 3.3: Inserting More Data into the patient Table

```
INSERT INTO APPOINTMENT_INFO (P_NO, D_NAME, SPECIALIZATION,
    APPOINTMENT_DATE)
VALUES ('P-101', 'MR. X', 'CS', TO_DATE('2023-08-25', 'YYYY-MM-DD'
    ));

INSERT INTO APPOINTMENT_INFO (P_NO, D_NAME, SPECIALIZATION,
    APPOINTMENT_DATE)
VALUES ('P-102', 'MR. Y', 'GS', TO_DATE('2023-08-26', 'YYYY-MM-DD'
    ));

INSERT INTO APPOINTMENT_INFO (P_NO, D_NAME, SPECIALIZATION,
    APPOINTMENT_DATE)
VALUES ('P-103', 'MR. Z', 'IM', TO_DATE('2023-08-27', 'YYYY-MM-DD'
    ));
```

Listing 3.4: Inserting More Data into the appointment Table

Chapter 4

Table Alter

4.0.1 Database Schema Modifications

In this section, we will explain a series of SQL queries that involve modifying a database schema.

4.0.2 Adding a Column

We start by adding a new column `APPOINTMENT_DATE` of type `DATE` to the `APPOINTMENT` table.

```
1 ALTER TABLE APPOINTMENT ADD APPOINTMENT_DATE DATE;
```

Listing 4.1: Adding a Column

4.0.3 Dropping a Constraint

Next, we drop an existing primary key constraint named `APPOINTMENT_PK` from the `APPOINTMENT` table.

```
1 ALTER TABLE APPOINTMENT DROP CONSTRAINT APPOINTMENT_PK;
```

Listing 4.2: Dropping a Constraint

4.0.4 Adding a New Primary Key Constraint

We then add a new primary key constraint APPOINTMENT_PK to the APPOINTMENT table with the columns P_NO, D_NAME, SPACIALIZATION, and APPOINTMENT_DATE.

```
1 ALTER TABLE APPOINTMENT
2 ADD CONSTRAINT APPOINTMENT_PK
3 PRIMARY KEY (P_NO, D_NAME, SPACIALIZATION, APPOINTMENT_DATE);
```

Listing 4.3: Adding a New Primary Key Constraint

4.0.5 Dropping a Primary Key Constraint Again

Strangely, the primary key constraint APPOINTMENT_PK is dropped again. This might be an error as it was added and dropped twice.

```
1 ALTER TABLE APPOINTMENT DROP CONSTRAINT APPOINTMENT_PK;
```

Listing 4.4: Dropping a Primary Key Constraint

4.0.6 Renaming Columns

Columns are renamed in the APPOINTMENT table. First, PATIENT_NO is renamed to P_NO, and then NAME is renamed to D_NAME.

```
1 ALTER TABLE APPOINTMENT RENAME COLUMN PATIENT_NO TO P_NO;
2 ALTER TABLE APPOINTMENT RENAME COLUMN NAME TO D_NAME;
```

Listing 4.5: Renaming Columns

4.0.7 Renaming the Table

The APPOINTMENT table is renamed to APPOINTMENT_INFO.

```
1 ALTER TABLE APPOINTMENT RENAME TO APPOINTMENT_INFO;
```

Listing 4.6: Renaming the Table

4.0.8 Adding a New Primary Key Constraint After Renaming

After renaming, a new primary key constraint `APPOINTMENT_PK` is added to the `APPOINTMENT_INFO` table with the renamed columns.

```
ALTER TABLE APPOINTMENT_INFO
ADD CONSTRAINT APPOINTMENT_PK
PRIMARY KEY (P_NO, D_NAME, SPACIALIZATION, APPOINTMENT_DATE);
```

Listing 4.7: Adding a New Primary Key Constraint After Renaming

4.0.9 Adding Foreign Key Constraints

Foreign key constraints are added to the `APPOINTMENT_INFO` table. First, a foreign key `FK_APPOINTMENT_P_NO` is added referencing the `PATIENT` table's `PATIENT_NO` column with cascading delete.

```
ALTER TABLE APPOINTMENT_INFO
ADD CONSTRAINT FK_APPOINTMENT_P_NO
FOREIGN KEY (P_NO) REFERENCES PATIENT (PATIENT_NO)
ON DELETE CASCADE;
```

Listing 4.8: Adding a Foreign Key Constraint

Another foreign key `FK_APPOINTMENT_D_NAME` is added referencing the `DOCTOR` table's `NAME` and `SPACIALIZATION` columns with cascading delete.

```
ALTER TABLE APPOINTMENT_INFO
ADD CONSTRAINT FK_APPOINTMENT_D_NAME
FOREIGN KEY (D_NAME, SPACIALIZATION)
REFERENCES DOCTOR (NAME, SPACIALIZATION)
ON DELETE CASCADE;
```

Listing 4.9: Adding Another Foreign Key Constraint

Chapter 5

Performing Select Query

5.1 SQL Query Explanations

In this section, we will explain a series of SQL queries.

5.1.1 Query A

Find all the Doctors' names whose fees are less than 1500.

```
1 SELECT NAME FROM DOCTOR WHERE FEE < 1500;
```

Listing 5.1: Query A

This query retrieves the names of doctors from the 'DOCTOR' table whose fees are less than 1500.

5.1.2 Query B

Find all the Patients' names who live in 'KHL' city.

```
1 SELECT NAME FROM PATIENT WHERE ADDRESS = 'KHL';
```

Listing 5.2: Query B

This query fetches the names of patients from the 'PATIENT' table who live in the city 'KHL'.

5.1.3 Query C

Show the result of Cartesian Product between PATIENT and APPOINTMENT_INFO table.

```
1 SELECT * FROM PATIENT, APPOINTMENT_INFO;
```

Listing 5.3: Query C

This query performs a Cartesian product between the 'PATIENT' and 'APPOINTMENT_INFO' tables, producing a result set with all possible combinations of rows from both tables.

5.1.4 Query D

Show the result of Natural Join between PATIENT and APPOINTMENT_INFO table.

```
1 SELECT * FROM PATIENT NATURAL JOIN APPOINTMENT_INFO;
```

Listing 5.4: Query D

This query performs a natural join between the 'PATIENT' and 'APPOINTMENT_INFO' tables, which matches rows based on columns with the same names in both tables.

5.1.5 Query E

Find all the Patient's names and their address who have an appointment today.

5.1.6 Query F

Find all the Doctor-related information who have patients from 'DHK'.

```
1 SELECT NAME, SPECIALIZATION, FEE FROM DOCTOR, APPOINTMENT_INFO
2 WHERE DOCTOR.NAME = APPOINTMENT_INFO.D_NAME AND APPOINTMENT_INFO.
     P_NO
3 IN (SELECT PATIENT_NO FROM PATIENT WHERE ADDRESS = 'DHK');
```

Listing 5.5: Query F

This query retrieves the names, specializations, and fees of doctors from the 'DOCTOR' table who have patients residing in the city 'DHK'. It achieves this by joining the 'DOCTOR' and 'APPOINTMENT_INFO' tables and filtering the results based on the specified conditions.

Chapter 6

Table Modifications

6.1 Update and Delete Operations

In this section, we will explain a series of SQL queries involving update and delete operations.

6.1.1 Query A

Update the NAME and ADDRESS of a tuple from 'A' and 'DHK' to 'K' and 'RAJ' accordingly.

```
1 UPDATE PATIENT
2 SET NAME = 'K', ADDRESS = 'RAJ'
3 WHERE NAME = 'A';
```

Listing 6.1: Updating Tuple NAME and ADDRESS

This query updates the 'NAME' and 'ADDRESS' columns of the 'PATIENT' table for the tuple where the 'NAME' is 'A' and changes it to 'K' and 'RAJ', respectively.

6.1.2 Query B

Update the NAME of table DOCTOR from 'MR. Y' to 'Ms. Y'.

```
1 UPDATE DOCTOR
```

```
2 SET NAME = 'Ms. Y'
3 WHERE NAME = 'MR. Y';
```

Listing 6.2: Updating DOCTOR Table NAME

This query updates the 'NAME' column of the 'DOCTOR' table where the name is 'MR. Y' and changes it to 'Ms. Y'.

6.1.3 Query C

Delete Patient with PATIENT_NO P-101.

```
1 DELETE FROM PATIENT
2 WHERE PATIENT_NO = 'P-101';
```

Listing 6.3: Deleting a Patient

This query deletes the row from the 'PATIENT' table where the 'PATIENT_NO' is 'P-101'.

6.1.4 Query D

Delete all the information without deleting the table structure.

```
1 DELETE FROM DOCTOR;
```

Listing 6.4: Deleting All Information

This query deletes all rows from the 'DOCTOR' table, effectively removing all the data without altering the table structure.