

Question 5. [8 MARKS]

Suppose I have a file called `nonsense.ddl` containing this:

```
DROP SCHEMA IF EXISTS rp CASCADE;
CREATE SCHEMA rp;
SET SEARCH_PATH TO rp;

CREATE TABLE Things (
    A INT PRIMARY KEY,
    B INT,
    C INT UNIQUE
);

CREATE TABLE Junk (
    G INT PRIMARY KEY,
    H INT,
    I INT,
    FOREIGN KEY (I) REFERENCES Things(A) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE Stuff (
    D INT,
    E INT,
    F INT PRIMARY KEY,
    FOREIGN KEY (E) REFERENCES Things(C) ON UPDATE RESTRICT ON DELETE SET NULL,
    FOREIGN KEY (E) REFERENCES Junk(G) ON UPDATE SET NULL ON DELETE CASCADE
);
```

Part (a) [2 MARKS]

Suppose I imported this file into postgresQL using the command `\i nonsense.ddl` and then a few weeks later the following happened when I tried to access table `Junk`.

```
dbsrv1% psql csc343h-dianeh
psql (9.1.15, server 9.1.14)
Type "help" for help.
```

```
csc343h-dianeh=> SELECT * FROM Junk;
ERROR:  relation "junk" does not exist
LINE 1: SELECT * FROM Junk;
          ^
```

Modify my interaction above so that the `SELECT` statement works.

Solutions:

The table Junk is still defined, but we haven't referred to it successfully. We can either give a fully qualified name for it:

```
csc343h-dianeh=> select * from rp.Junk;
 g | h | i
---+---+---
 3 | 2 | 9
 6 | 2 | 8
 8 | 5 | 9
 4 | 1 | 1
(4 rows)
```

or we can set the search path so that we don't have to:

```
csc343h-dianeh=> set search_path to rp;
SET
csc343h-dianeh=> select * from Junk;
 g | h | i
---+---+---
 3 | 2 | 9
 6 | 2 | 8
 8 | 5 | 9
 4 | 1 | 1
(4 rows)
```

Part (b) [2 MARKS]

What is the most important thing that is the same about PRIMARY KEY and UNIQUE?

Solutions:

For both, there can be **no duplicates**. That is, whether a set of attributes a_1, a_2, \dots, a_n is PRIMARY KEY or UNIQUE, there can be no two tuples with the same value for a_1 , and the same value for a_2 and ... and the same value for a_n .

What is one important difference between PRIMARY KEY and UNIQUE?

Solutions:

A table can declare any number of sets of attributes UNIQUE, but it can only have one PRIMARY KEY. Another difference is that the DBMS will / is highly likely to make an index on a PRIMARY KEY, but may not choose to do so for a set of attributes that is merely declared to be UNIQUE.

Part (c) [2 MARKS]

Suppose the tables have been populated as shown below. Modify the data to show the contents of the three tables after this command is executed:

```
UPDATE Things SET C = 20 WHERE A = 8;
```

Things:	Stuff:	Junk:
a b c	d e f	g h i
---+---+---	---+---+---	---+---+---
3 2 3	3 4 1	9 0 3
4 2 5	1 6 3	3 2 9
8 2 6 20	2 9 5	6 2 8
1 5 4	2 3 4	8 5 9
9 8 7		4 1 1
2 2 9		

Solutions:

There is no change to the tables, because the update is rejected:

```
csc343h-dianeh=> UPDATE Things SET C = 20 WHERE A = 8;    6 still references 6 so update rejected
ERROR:  update or delete on table "things" violates foreign key constraint "stuff_e_fkey"
        on table "stuff"
DETAIL:  Key (c)=(6) is still referenced from table "stuff".
```

update cascade, delete cascade

Part (d) [2 MARKS]

Suppose we began with the same original tables, shown below, but ran a different command. Modify the data to show the contents of the three tables after this command is executed:

DELETE FROM Things WHERE C = 3;

update set null, delete cascade

update restrict, delete set null

Things:

a	b	c
3	2	3
4	2	5
8	2	6
1	5	4
9	8	7
2	2	9

delete

Stuff:

d	e	f
3	4	1
1	6	3
2	9	5
2	3	4

delete cascade

3 -> null

Junk:

g	h	i
9	0	3
3	2	9
6	2	8
8	5	9
4	1	1

delete cascade

Solutions:

csc343h-dianeh=> DELETE FROM Things WHERE C = 3;

DELETE 1

csc343h-dianeh=> select * from Things;

a	b	c
4	2	5
8	2	6
1	5	4
9	8	7
2	2	9

(5 rows)

csc343h-dianeh=> select * from Stuff;

d	e	f
3	4	1
1	6	3
2		4

(3 rows)

csc343h-dianeh=> select * from Junk;

g	h	i
3	2	9
6	2	8
8	5	9
4	1	1

(4 rows)