Assignment-7 Solutions

GRADED QUESTIONS:

Problem 1.

Consider a relation PC (P, C) which indicates that person P is a parent of person C. Furthermore, assume that there are two unary relations Male(P) and Female(P) that specify the gender of a person P.

Write a program that defines the predicate Ancestor_Male_Female (x, y, z) which specifies that x is an ancestor of a male descendant y, and y is an ancestor of a female descendant z

```
create or replace function Ancestor_Male_Female(x integer, y integer, z integer)
returns boolean as
$$
  select exists (select a1.A, a1.D, a2.D
                 from ANC a1, ANC a2
                 where a1.A = x
                 and a1.D = y
                 and a2.A = a1.D
                 and a2.D = z
                 and a1.D in (select P from male)
                 and a2.D in (select P from female));
$$ language sql;
```

Problem 2.

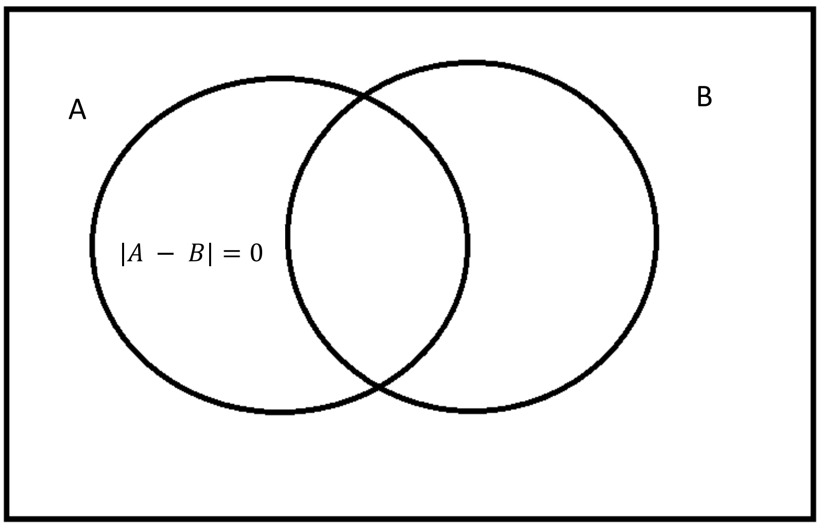
Formulate the following query using the Venn diagram without counting condition. Create function or views to represent the sets used in the query.

Find the pairs (p1, p2) of different person pids such that the person with pid p1 and the person with pid p2 knows the same number of persons.

```
CREATE OR REPLACE FUNCTION peopleKnown(pid integer)
RETURNS TABLE(total int) AS
$$
SELECT SUM(CASE WHEN k.pid1 IS NOT NULL THEN 1 ELSE 0 END) AS total
FROM knows k
WHERE k.pid1 = pid
$$ LANGUAGE SQL;
```

```sql
SELECT p1.pid, p2.pid
FROM person p1, person p2
WHERE p1.pid <> p2.pid
AND NOT EXISTS (SELECT total
        FROM peopleKnown(p1.pid)
        EXCEPT
        SELECT total
        FROM peopleKnown(p2.pid));
```

<u>Set A</u>: Number of persons known by person 1
<u>Set B</u>: Number of persons known by person 2



Problem 3.

Formulate the following query in object relational SQL queries. Use the relations, views, set operations and set predicates defined in assignment 5 with the same restrictions, i.e. You cannot use the Knows, companyLocation, and personSkill relations.

Find the pid of each person who has the most job skills

```sql
CREATE or REPLACE VIEW personHasSkills AS
SELECT DISTINCT p.pid,ARRAY(SELECT s.skill
                            FROM personSkill s
                            WHERE s.pid = p.pid
                            ORDER BY 1) AS skills
FROM Person p ORDER BY 1;
```

```sql
WITH skill_counts AS
(SELECT pid, cardinality(skills) AS num_skills
FROM personHasSkills),

max_skill_count AS
(SELECT max(num_skills) AS max_skills
FROM skill_counts)

SELECT pid
FROM skill_counts, max_skill_count
WHERE num_skills = max_skills;
```

Problem 4.

Consider two relations R(A, B) and S(B, C), two constant a and c, and a view with the following definition:

SELECT r.A, s.C
FROM R r, S s
WHERE r.A != a AND r.B = s.B AND s.C != c

Write a trigger that maintains the number of tuples in this view.

```sql
CREATE TABLE IF NOT EXISTS R(A INT, B INT);
CREATE TABLE IF NOT EXISTS S(B INT, C INT);
CREATE TABLE IF NOT EXISTS V(A INT, C INT);
CREATE TABLE count_v(total integer);
INSERT INTO count_v VALUES(0);

CREATE OR REPLACE FUNCTION count_func()
RETURNS trigger AS
$$
BEGIN
UPDATE count_v SET total = total + 1;
RETURN NULL;
END;
$$ LANGUAGE 'plpgsql';
CREATE TRIGGER total
AFTER INSERT ON V
FOR EACH ROW
EXECUTE PROCEDURE count_func();
```

Problem 5.

Consider the relations R(A, B), S(B,C), and T(C, D). Assume that R, S, and T are stored in B(R), B(S), and B(T) blocks, respectively. Furthermore, assume that you have a buffer of (approximate) size M

Assuming that you use the block nested-loop join algorithm to implement natural join operations, specify the time complexity to evaluate the relational algebra expression $(R \bowtie S) \bowtie T$. You can make the assumption that $B(R \bowtie S) \leq M2$, where $B(R \bowtie S)$ is the number of blocks to store $(R \bowtie S)$

$$B(R \bowtie S) = B(R) + \frac{B(R) \times B(S)}{M}$$

$$B((R \bowtie S) \bowtie T) = B(R \bowtie S) + \frac{B(R \bowtie S) \times B(T)}{M}$$

Problem 6.

Suppose that we have an ordered file with r = 300,000 records stored on a disk with block size B = 4,096 bytes. The length of the record is 100 bytes.

(a) Compute the number of block accesses required to search for a record.

No of records: r = 300,000
Block size: B = 4096 bytes
Record length: R = 100 bytes
index length: i = 15 bytes

No of records per block: rB = floor(B/R) = floor(4096/100) = 40 records
No of blocks in the sequential file: bF = floor(r/rB) = floor(7500) = 7500 blocks
If the file is not sorted, then a linear search needs to be performed
No of block access to search for a record = 7500/2 = 3750 block accesses (on average)
If the file is sorted, binary search can be performed
No of block access for binary search: ceiling(log_2(bF)) = ceiling(12.87) = 13 block accesses

(b) Now suppose that the ordering key field of the file is V = 9 bytes long, a block pointer is P = 6 bytes long, and we have constructed a primary index for the file. Compute the number of block accesses required to search for a record using the primary index.

We need to search for a block in the index instead of the file
No of index entries per block: iB = floor(B/i) = floor(4096/15) = 273 index entries
No of blocks in the index file: bI = floor(bF/iB) = ceiling(7500/273) = ceiling(27.47) = 28 blocks
Since the index is sorted, binary search can be performed
No of block access in the index file: ceiling(log_2(bI)) = ceiling(4.8) = 5 block accesses

Problem 7.

Let x, y, and z be data objects. State which of the following schedules are conflict-serializable or not conflict-serializable, and for each schedule that is serializable, give a serial schedule with which that schedule is conflict-equivalent

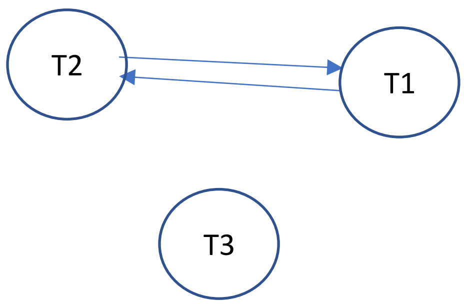
(a) R1(x); R2(y); R1(z); R2(x); R1(y)

Conflict serializable as there are no conflicting actions (all reads).

The serial schedule is as follows:

R1(x) R1(z) R1(y) R2(z) R2(x)

(b) R1(x); W2(y); R1(z); R3(z); W2(x); R1(y)



This is a cyclic graph; therefore, it is not conflict serializable and we cannot have a serial schedule for it.

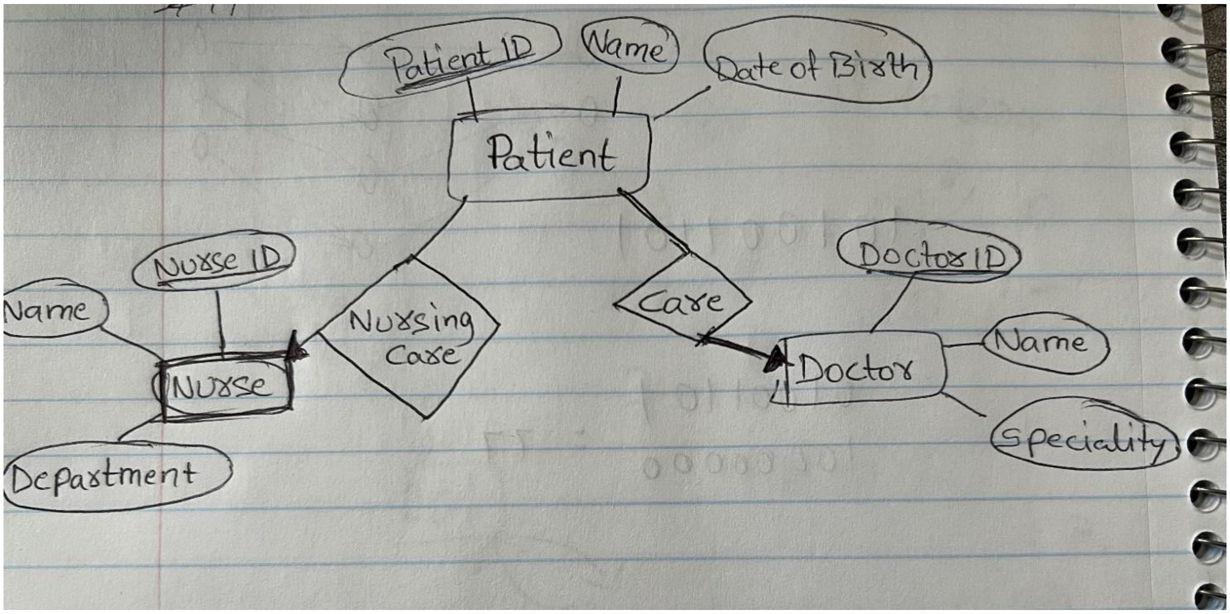
Problem 8.

Scenario: A hospital manages patients, doctors, and nurses. Each patient has a unique patient ID, a name, and a date of birth. Each doctor has a unique doctor ID, a name, and a specialty. Each nurse has a unique nurse ID, a name, and a department.
Patients can be assigned to doctors for care. Each doctor can care for multiple patients. Nurses can be assigned to patients for care. Each nurse can care for multiple patients.

a) Create an Entity Relationship Diagram for the above scenario.

b) Convert the above entity relationship diagram to schema.

a.



b. CREATE TABLE Patient(patientID integer PRIMARY KEY, name text, dateOfBirth text);
CREATE TABLE Doctor(doctorID integer PRIMARY KEY, name text, specialty text);
CREATE TABLE Nurse(nurseID integer PRIMARY KEY, name text, department text);
CREATE TABLE Care(patientID REFERENCES Patient(patientID), doctorID REFERENCES Doctor(doctorID));
CREATE TABLE NurseCare(patientID REFERENCES Patient(patientID), nurseID REFERENCES Nurse(nurseID));

PRACTICE QUESTIONS:

Problem 1.

Write a PL/pgSQL function that takes an integer as an input and returns the sum of first n prime numbers.

```
CREATE OR REPLACE FUNCTION sum_of_primes(n INTEGER) RETURNS INTEGER AS $$
DECLARE
    prime_count INTEGER := 0;
    current_number INTEGER := 2;
    sum_of_primes INTEGER := 0;
BEGIN
    WHILE prime_count < n LOOP
        DECLARE
            is_prime BOOLEAN := TRUE;
        BEGIN
            FOR i IN 2..(current_number/2) LOOP
                IF current_number % i = 0 THEN
                    is_prime := FALSE;
                    EXIT;
                END IF;
            END LOOP;
            IF is_prime THEN
                prime_count := prime_count + 1;
                sum_of_primes := sum_of_primes + current_number;
            END IF;
            current_number := current_number + 1;
        END;
    END LOOP;
    RETURN sum_of_primes;
END;
$$ LANGUAGE plpgsql;
```

Problem 2.

Write a PL/pgSQL function that sorts a given array using selection sort

```
CREATE OR REPLACE FUNCTION selection_sort(arr INTEGER[]) RETURNS INTEGER[] AS $$
DECLARE
    n INTEGER := array_length(arr, 1);
BEGIN
    FOR i IN 1..n-1 LOOP
        DECLARE
            min_idx INTEGER := i;
```

```
        BEGIN
            FOR j IN i+1..n LOOP
                IF arr[j] < arr[min_idx] THEN
                    min_idx := j;
                END IF;
            END LOOP;
            IF min_idx != i THEN
                arr[i], arr[min_idx] := arr[min_idx], arr[i];
            END IF;
        END;
    END LOOP;
    RETURN arr;
END;
$$ LANGUAGE plpgsql;
```

Problem 3.

Formulate the following query using the Venn diagram without counting condition. Create function or views to represent the sets used in the query.

Find the cname of each company who only employs persons who make less than 50000.

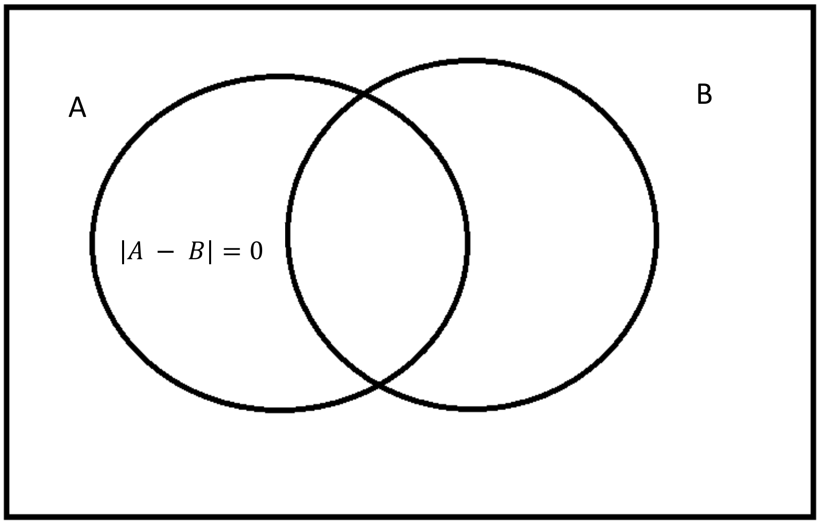
```
CREATE OR REPLACE FUNCTION All_Employees(companyName text)
returns table (pid integer) AS
$$
SELECT DISTINCT w.pid
FROM worksFor w
WHERE w.cname = companyName
$$ language sql;

CREATE OR REPLACE FUNCTION Employeesless50000(companyName text)
returns table (pid integer) AS
$$
SELECT DISTINCT w.pid
FROM worksFor w
WHERE w.cname = companyName
AND w.salary<50000
$$ language sql;

SELECT c.cname
FROM Company c
WHERE NOT EXISTS(SELECT pid
FROM All_Employees(c.cname)
EXCEPT
SELECT pid
```

```
FROM Employeesless50000(c.cname));
```

Set A: Pid of all persons of a particular company
Set B: Pid of persons of a particular company who has salary less than 50000



Problem 4.

Formulate the following query in object relational SQL query. Use the relations, views, set operations and set predicates defined in assignment 6 with the same restrictions, i.e. You cannot use the Knows, companyLocation, and personSkill relations.

Find, for each person, that person's pid and name along with the number of persons he or she manages.

Ans.
```
SELECT p.pid, p.pname, (SELECT COUNT(hm.eid)
                        FROM hasManager hm
                        WHERE hm.mid = p.pid)
    FROM Person p;
```

Problem 5.

Consider the relations R(A, B), S(B,C), and T(C, D). Assume that R, S, and T are stored in B(R), B(S), and B(T) blocks, respectively. Furthermore, assume that you have a buffer of (approximate) size M

Assuming that you use the sort-merge join algorithm to implement natural join operations, specify the time complexity to evaluate the relational algebra expression $(R \bowtie S) \bowtie T$. You can make the assumption that $B(R \bowtie S) \leq M2$, where $B(R \bowtie S)$ is the number of blocks to store $(R \bowtie S)$

$$B(R \bowtie S) = B(R) + B(S) + 2B(R)\left\lceil log_M\left(B(R)\right)\right\rceil + 2B(S)\lceil log_M(B(S))\rceil$$

$$B\left((R \bowtie S) \bowtie R\right) = B(R \bowtie S) + B(T) + 2B(R \bowtie S)\left\lceil log_M\left(B(R \bowtie S)\right)\right\rceil + 2B(T)\lceil log_M(B(T))\rceil$$

Problem 6.

Suppose that we have an ordered file with r = 300,000 records stored on a disk with block size B = 4,096 bytes. The length of the record is 100 bytes. Suppose we want to search for a record with a specific value for a secondary key—a nonordering key field of the file that is V = 9 bytes long.

a) Compute the number of block accesses required to search for a record using the secondary key.

   Number of blocks required = (300,000 * 100) / 4096 = 7324 approximately

   Number of block access required = log2(7324) = 13 approx.

b) Suppose that we construct a secondary index on that nonordering key field of the file. Compute the number of block accesses required to search for a record using the secondary index.

   size of secondary index = 300,000 * 9 = 2,700,000
   no of index blocks = 2,700,000 / 4096 = 659
   no of block access required = log2(659) = 10

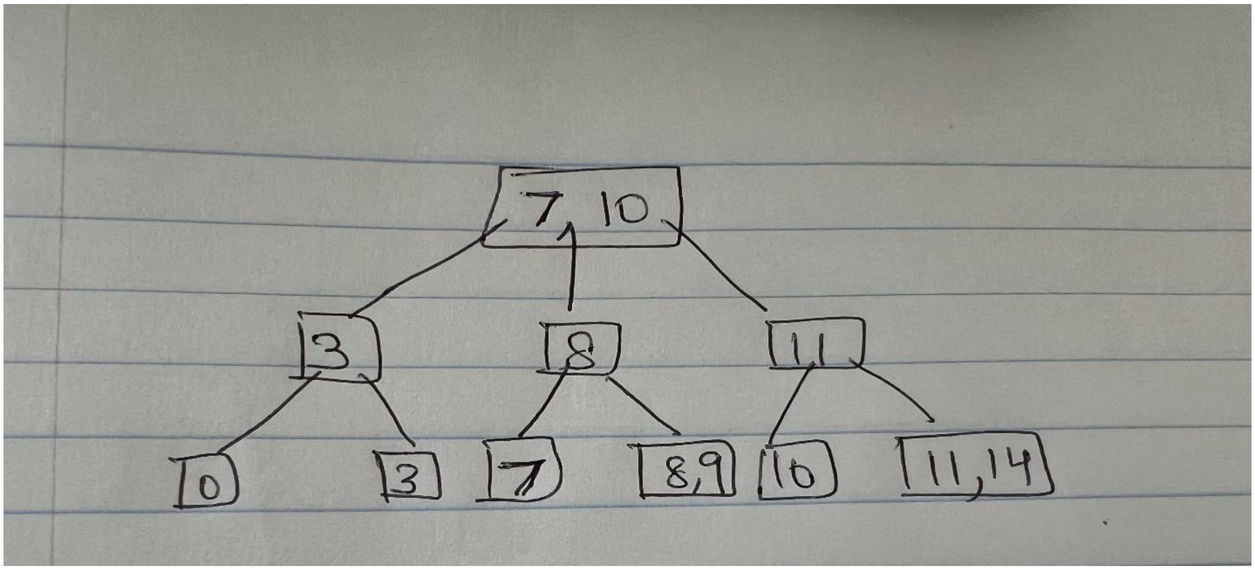
Problem 7.

Consider the following B+-tree of order n=2 that indexes records, with keys 3, 8, 10, and 11

```
-----------
| / 10 \ |
-----------
/ \
/ \
---------- ------------
| 3 8 |--> | 10 11 |
---------- ------------
```

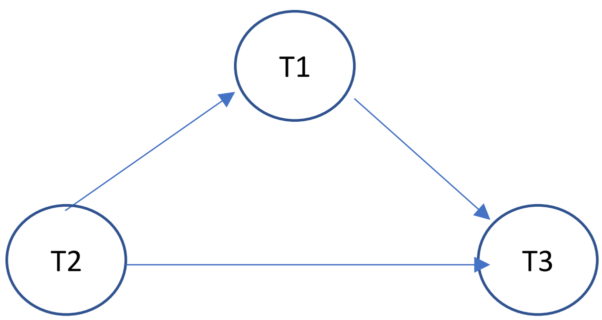
Show the contents of your B+-tree index after inserting records with keys 0, 7, 14, and 9 in that order.

Ans.



Problem 8.

R1(z); W2(x); R2(z); R2(y); W1(x); W3(z); W1(y); R3(x)

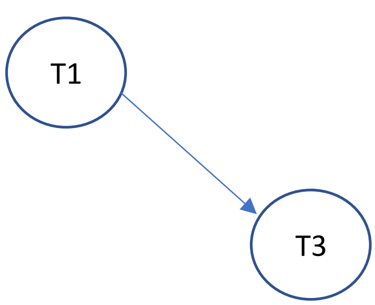


Conflict serializable as the graph is acyclic

Moving R2 to the left, as it has no incoming arcs

R2(z); R2(y); R1(z); w2(x); w1(x); w3(z); w1(y); R3(x)

Removing R2 from the graph



Moving R1 to the left

R2(z); R2(y); R1(z); w1(x); w1(y); w2(x); w3(z); R3(x)

Above schedule is serial