Stony Brook University

SUNY

Midterm Solutions

CSE532: Theory of Database Systems

Fusheng Wang

Department of Biomedical Informatics

Department of Computer Science

Problem 1

```
CREATE TABLE CSE532.PATIENT_DIAGNOSIS (
encounter_id INT NOT NULL,
patient_id INT,
diagnosis VARCHAR(64),
hospital_id INT,
visit_date DATE,
PRIMARY KEY (encounter_id)
);

CREATE TABLE CSE532.PATIENT_LOCATION (
patient_id INT NOT NULL,
address VARCHAR (64),
location DB2GSE.ST_POINT,
PRIMARY KEY (patient_id)
);

CREATE TABLE CSE532.HOSPITAL_LOCATION (
hospital_id INT NOT NULL,
address VARCHAR (64),
location DB2GSE.ST_POINT,
PRIMARY KEY (hospital_id)
);
```

Query 1: Sample Data

```
INSERT INTO CSE532.PATIENT_DIAGNOSIS
VALUES (10000, 2000, 'Acute myocardial infarction', 10, '2015-01-01');

INSERT INTO CSE532.PATIENT_DIAGNOSIS
VALUES (10001, 2000, 'Acute myocardial infarction', 20, '2015-02-01');

INSERT INTO CSE532.PATIENT_DIAGNOSIS
VALUES (10002, 2000, 'Acute myocardial infarction', 10, '2015-02-02');

INSERT INTO CSE532.PATIENT_DIAGNOSIS
VALUES (10003, 2000, 'Acute myocardial infarction', 20, '2015-03-02');


INSERT INTO CSE532.PATIENT_DIAGNOSIS
VALUES (20000, 3000, 'Asthma', 20, '2015-01-01');

INSERT INTO CSE532.PATIENT_DIAGNOSIS
VALUES (20001, 3000, 'Asthma', 10, '2015-02-01');

INSERT INTO CSE532.PATIENT_DIAGNOSIS
VALUES (20002, 3000, 'Asthma', 30, '2015-02-02');
```

Query 1: Example Pairs

```
SELECT A.hospital_id, B.hospital_id,
       COUNT(DISTINCT A.patient_id) AS overlap_count
FROM CSE532.PATIENT_DIAGNOSIS A, CSE532.PATIENT_DIAGNOSIS B
WHERE A.patient_id = B.patient_id AND
       A.hospital_id != B.hospital_id
GROUP BY A.hospital_id, B.hospital_id
ORDER BY overlap_count desc;
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results | Progress | Access Plan Diagram

	HOSPITAL_ID	HOSPITAL_ID	OVERLAP_COUNT
1	10	20	2
2	20	10	2
3	10	30	1
4	20	30	1
5	30	10	1
6	30	20	1

Query 1: Method 1

```
SELECT A.hospital_id, B.hospital_id,
       COUNT(DISTINCT A.patient_id) AS overlap_count
FROM CSE532.PATIENT_DIAGNOSIS A, CSE532.PATIENT_DIAGNOSIS B
WHERE A.patient_id = B.patient_id AND
       A.hospital_id > B.hospital_id
GROUP BY A.hospital_id, B.hospital_id
ORDER BY overlap_count desc
FETCH FIRST ROW ONLY;
```

	HOSPITAL_ID	HOSPITAL_ID	OVERLAP_COUNT
1	20	10	2

Query 1: Method 2: Two Common Table Expressions

```
WITH OVERLAP_PAIR AS (
     SELECT A.hospital_id AS a_id, B.hospital_id b_id,
          COUNT(DISTINCT A.patient_id) AS overlap_count
     FROM CSE532.PATIENT_DIAGNOSIS A, CSE532.PATIENT_DIAGNOSIS B
     WHERE A.patient_id = B.patient_id AND
          A.hospital_id > B.hospital_id
     GROUP BY A.hospital_id, B.hospital_id
),

DENSE_RANK AS (
     SELECT a_id, b_id,
     RANK() OVER (ORDER BY overlap_count DESC) AS rank
     FROM OVERLAP_PAIR
)

SELECT a_id, b_id
FROM DENSE_RANK
WHERE rank = 1;
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results | Progress | Access Plan Diagram

	A_ID	B_ID
1	20	10

Query 2: Example Records

```
INSERT INTO CSE532.HOSPITAL_LOCATION VALUES(
10, '101 Nicolls Rd, Stony Brook, NY 11794',
DB2GSE.ST_POINT(-73.140943, 40.925654, 1)
);

INSERT INTO CSE532.HOSPITAL_LOCATION VALUES(
20, 'E 101st St, New York, NY 10029',
DB2GSE.ST_POINT(-73.947804, 40.788722, 1)
);

INSERT INTO CSE532.PATIENT_LOCATION VALUES(
2000, '1 Presidents Dr APT 4B, Port Jefferson, NY 11777',
DB2GSE.ST_POINT(-73.070906, 40.935913, 1)
);

INSERT INTO CSE532.PATIENT_LOCATION VALUES(
3000, '9 Williams Blvd, Lake Grove, NY 11755',
DB2GSE.ST_POINT(-73.111875, 40.856730, 1)
);
```

Stony Brook University Hospital

Mount Sinai Hospital (Manhattan)

```
SELECT ploc.patient_id, hloc.hospital_id,
DB2GSE.ST_DISTANCE (hloc.location, ploc.location, 'STATUTE MILE') AS dist
FROM CSE532.HOSPITAL_LOCATION AS hloc, CSE532.patient_location ploc;
```

	PATIENT_ID	HOSPITAL_ID	DIST
1	2000	10	3.7330407455552272
2	3000	10	4.993738080994404
3	2000	20	47.04792321213736
4	3000	20	44.06893420939502

Query 2: Method 1: Brute Force

```
WITH HOSPTIAL_DIST AS (
   SELECT ploc.patient_id, hloc.hospital_id,
   DB2GSE.ST_DISTANCE (hloc.location, ploc.location, 'STATUTE MILE') AS dist
   FROM CSE532.HOSPITAL_LOCATION AS hloc, CSE532.patient_location ploc
  ),
  DIST_RANK AS (
  SELECT patient_id, hospital_id,
  RANK () OVER (PARTITION BY patient_id ORDER BY dist ASC) AS rank
  FROM HOSPTIAL_DIST
  )
 SELECT distinct r.patient_id,
        CASE  d.patient_id WHEN null THEN 'false'
        ELSE 'true' END AS visited
 FROM DIST_RANK r LEFT OUTER JOIN CSE532.PATIENT_DIAGNOSIS d
    ON r.patient_id = d.patient_id
    AND r.rank = 1;
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results | Progress | Access Plan Diagram

	PATIENT_ID	VISITED
1	2000	true
2	3000	true

Query 2: Method 2

```
WITH HOSPTIAL_DIST AS (
  SELECT ploc.patient_id, hloc.hospital_id,
   DB2GSE.ST_DISTANCE (hloc.location, ploc.location, 'STATUTE MILE') AS dist
  FROM CSE532.HOSPITAL_LOCATION AS hloc, CSE532.patient_location ploc
  WHERE DB2GSE.ST_CONTAINS(DB2GSE.ST_BUFFER(ploc.location, 5,  'STATUTE MILE'),
        hloc.location) = 1
 ),
 DIST_RANK AS (
  SELECT patient_id, hospital_id,
  RANK () OVER (PARTITION BY patient_id ORDER BY dist ASC) AS rank
  FROM HOSPTIAL_DIST
 )
SELECT distinct r.patient_id,
       CASE  d.patient_id WHEN null THEN 'false'
       ELSE 'true' END AS visited
FROM DIST_RANK r LEFT OUTER JOIN CSE532.PATIENT_DIAGNOSIS d
   ON r.patient_id = d.patient_id
   AND r.rank = 1;
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results | Progress | Access Plan Diagram

	PATIENT_ID	VISITED
1	2000	true
2	3000	true

Problem 2

- We have a table employee(empno, name, workdeptno, salary), where empno is the primary key. Assume the primary key is clustered, and the index is cached. The database uses a hard drive with 15K rpm, transfer rate 250MB/second, and seek time 3ms.
- Suppose the table has 1 million records, each record has 100 bytes, and the page size is 4KB. Estimate the cost for the query below. We assume 1 percent of the records are returned.
- SELECT * FROM employee WHERE empno >= 100000 and empno <= 110000

Problem 2

- Since the data is clustered on empno, and the search key is empno, we assume the data is stored contiguously.

The total size of records to be read:

1M x 100 x 0.01 bytes = 1M bytes

Seek time: 3ms

Rotational delay = 60/15000/2 = 0.002 seconds = 2ms

Data transfer time: 1M/(250x1024x1024) = 0.0038 seconds = 3.8 ms

Total time = seek time + rotational delay + data transfer time = 3 + 2 + 3.8 = 8.8ms

a

The following statements are true for database management systems:

- Data independence means that a DBMS should not be impacted by changing of applications.
- A table of RDBMS is a logical representation of the data, and the order of records in a table is the same as insertion order.
- ✓ When reading a record from a database, the smallest unit of data is a page.

b

If we want to use fixed grid index for large scale geospatial database, such as a map database of all geospatial objects of the world, the following statements are true:

- The grid index is not suitable as queries will involve too many geometric computations.
- ✓ Serious page overflow could happen and search could be very slow.

C

Comparing a client side JDBC program with a server side stored procedure for implementing a stand deviation computation for a large table, the following statements are true:

- JDBC program runs faster as it can use much more memory from the client machine.
- ✓ A stored procedure can run faster as there is no network data transfer cost.

d

Comparing a B+-Tree index with an R-Tree index, the following statements are true:

- Both indexes use a total order to sort keys.
- ✓ A point search (find all containing objects for a given point) in an R-Tree could traverse multiple child branches of a node in the tree.
- A single key search in a B+-Tree could traverse multiple child branches of a node in the tree.

e

The following statements are true for ranking or OLAP queries:

- For a top K query to return top K paid employees, FETCH FIRST K RECORDS can return the answer but runs slower compared to a RANK based method based on descending sorted salaries.
- An OLAP query with ROLLUP (year, country) is the same as ROLLUP (country, year).
- ✓ An OLAP query with CUBE (year, country) is the same as CUBE (country, year).