SCIT
School of Computing and Information Technology

Family Name	....
First Name	....
Student Number	....

CSCI317
Database Performance Tuning

This paper is for students studying at the Singapore Institute of Management Pte Ltd.

S1-2019 FINAL EXAMINATION

Date: ???

Time: ???

Exam value: **60% of the subject assessment.**

Marks available: **60 marks.**

DIRECTIONS TO CANDIDATES

1. The examination paper is printed on **both** sides.
2. All answers must be written in the answer booklet provided.
3. Distinct parts should be started on distinct pages.
4. In case of conflict, instructions here override answer booklet instructions.

EXAMINATION MATERIALS/AIDS ALLOWED

Nil

THIS EXAMINATION PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

VERSION 1

Introduction

The questions 2, 4, 5, 6, and 7 of the examination paper are related to the following simplified version of TPC-HR benchmark database used in the laboratory classes.

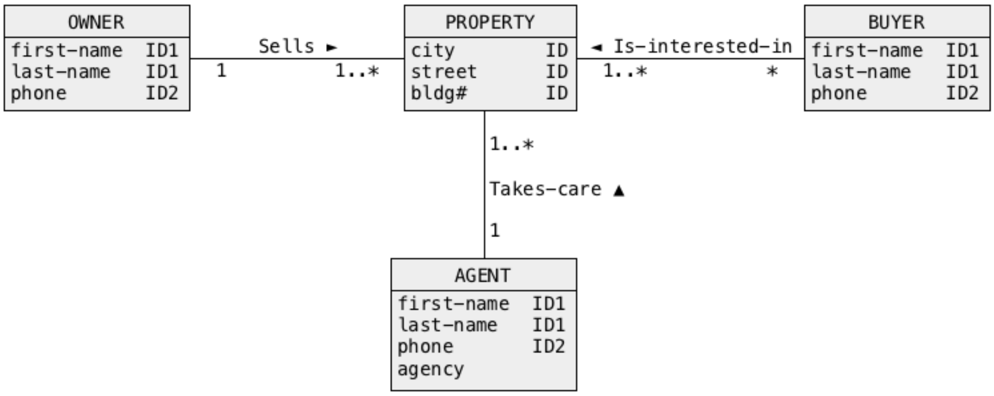
```
CUSTOMER(  C_CUSTKEY        NUMBER(12)       NOT NULL,
           C_NAME           VARCHAR(25)      NOT NULL,
           C_ADDRESS        VARCHAR(40)      NOT NULL,
           C_NATIONKEY      NUMBER(12)       NOT NULL,
      CONSTRAINT CUSTOMER_PKEY PRIMARY KEY(C_CUSTKEY) );

PART(      P_PARTKEY        NUMBER(12)       NOT NULL,
           P_NAME           VARCHAR(55)      NOT NULL,
           P_BRAND          CHAR(10)         NOT NULL,
           P_SIZE           NUMBER(12)       NOT NULL,
           P_RETAILPRICE    NUMBER(12,2)     NOT NULL,
      CONSTRAINT PART_PKEY PRIMARY KEY (P_PARTKEY) );

PARTSUPP(  PS_PARTKEY       NUMBER(12)       NOT NULL,
           PS_SUPPNAME      VARCHAR(55)      NOT NULL,
           PS_AVAILQTY      NUMBER(12)       NOT NULL,
      CONSTRAINT PARTSUPP_PKEY PRIMARY KEY (PS_PARTKEY,PS_SUPPNAME),
      CONSTRAINT PARTSUPP_FKEY FOREIGN KEY(PS_PARTKEY)
               REFERENCES PART(P_PARTKEY) );

ORDERS(    O_ORDERKEY       NUMBER(12)       NOT NULL,
           O_CUSTKEY        NUMBER(12)       NOT NULL,
           O_TOTALPRICE     NUMBER(12,2)     NOT NULL,
           O_ORDERDATE      DATE             NOT NULL,
      CONSTRAINT ORDERS_PKEY PRIMARY KEY (O_ORDERKEY),
      CONSTRAINT ORDERS_FKEY1 FOREIGN KEY (O_CUSTKEY)
                REFERENCES CUSTOMER(C_CUSTKEY) );

LINEITEM(  L_ORDERKEY       NUMBER(12)       NOT NULL,
           L_PARTKEY        NUMBER(12)       NOT NULL,
           L_LINENUMBER     NUMBER(12)       NOT NULL,
           L_QUANTITY       NUMBER(12,2)     NOT NULL,
           L_SHIPDATE       DATE             NOT NULL,
     CONSTRAINT LINEITEM_PKEY PRIMARY KEY (L_ORDERKEY, L_LINENUMBER),
     CONSTRAINT LINEITEM_FKEY1 FOREIGN KEY (L_ORDERKEY)
                REFERENCES ORDERS(O_ORDERKEY),
     CONSTRAINT LINEITEM_FKEY2 FOREIGN KEY (L_PARTKEY)
                REFERENCES PART(P_PARTKEY) );
```

Assume that, the relational tables listed above occupy the following amounts of disk storage:

```
CUSTOMER        100 Mbytes
PART             30 Mbytes
PARTSUPP        400 Mbytes
ORDERS          500 Mbytes
LINEITEM        900 Mbytes
```

Question 1 (7 marks)

The following conceptual schema represents a database domain where owners sell real estate properties, buyers are interested in real estate properties and sellers take care about real estate properties.



An objective of this task is to use <u>denormalization</u> to improve the performance of the following class of applications.

> *Find the phones of buyers (attributes phone in a class `BUYER`) who are interested in the real estate properties located in a given city (attribute city in a class `PROPERTY`) and being taken care about by an agent from a given agency (attribute agency in a class `AGENT`)*

A sample application that belongs to a class described above could be the following.

> *Find the phones of buyers who are interested in the real estate property located in Sydney and being taken care about by an agent from an agency Real Estate Demolishers.*

(1) Perform simplification of a conceptual schema given above and redraw a simplified schema.

(2 marks)

(2) To improve performance of a class of database applications given above denormalize a conceptual schema obtained in step (1) and redraw a denormalized schema.

(6 marks)

Question 2 (10 marks)

For each one of `SELECT` statements listed below find an index that speeds up the processing of a statement in the best possible way. Note, that an index must be created separately for each one of `SELECT` statements. Write `CREATE INDEX` statements to create the indexes.

(1)
```
SELECT P_BRAND, COUNT(*)
FROM PART
GROUP BY P_BRAND
HAVING COUNT(*) > 2;
```
(2 marks)

(2)
```
SELECT AVG(L_QUANTITY)
FROM ORDERS JOIN LINEITEM
           ON O_ORDERKEY = L_ORDERKEY;
```
(2 marks)

(3)
```
SELECT AVG(OPS_AVAILQTY)
FROM PARTSUPP
WHERE PS_SUPPNAME = 'James';
```
(2 marks)

(4)
```
SELECT P_NAME,
FROM PART
WHERE P_BRAND = 'RUBBISH'
ORDER BY P_NAME;
```
(2 marks)

(5)
```
SELECT C_NAME
FROM CUSTOMER
WHERE C_NATIONKEY = 'SG';
MINUS
SELECT C_NAME
FROM CUSTOMER
WHERE C_ADDRESS LIKE '%Bugis%';
```
(2 marks)

Question 3 (10 marks)

Assume that a relational table

```
PRODUCT(name, manufacturer, price, description, quality, mdate)
```

contains information about the names, manufacturers, prices, qualities, manufacturing dates and descriptions of products. Assume that the table has a composite primary key (name, manufacturer).

A database administrator created B*-Tree index on an attribute price. B*-tree index on a primary key has been automatically created by a database system.

Assume that:

(i) a relational table PRODUCT occupies 10^4 data blocks,
(ii) a relational table PRODUCT contains 10^5 rows,
(iii) a height of an index on the primary key is equal to 4,
(iv) a height of an index on an attribute price is equal to 2,
(v) the total number of distinct values in a column price is equal to 10^3,
(vi) a leaf level of an index on the primary key consists of 500 data blocks,
(vii) a leaf level of an index on attribute price consists of 100 data blocks.

List the comprehensive descriptions of query processing plans for each one of the queries listed below and estimate the total number of read block operations needed to compute each one of the queries (**show all calculations):**

(1)
```
SELECT price, COUNT(*)
FROM PRODUCT
GROUP BY price;
```
(2 marks)

(2)
```
SELECT name, manufacturer
FROM PRODUCT
WHERE manufacturer = 'IBM' or name = 'PC';
```
(2 marks)

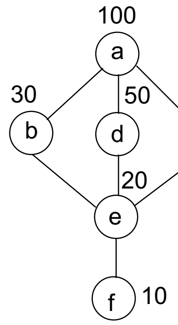
(3)
```
SELECT DISTINCT price
FROM PRODUCT
ORDER BY price;
```
(2 marks)

(4)
```
SELECT *
FROM PRODUCT
WHERE manufacturer = 'IBM' or name = 'PC';
```
(2 marks)

(5)
```
SELECT *
FROM PRODUCT
WHERE price = 100;
```
(2 marks)

Question 4 (8 marks)

Consider the following incomplete lattice of materialized views.



Assume that a view "a" has been already materialized and a cost of its materialization is 100. The costs of materialization of the other views "b", "c", "d", "e", and "f" are given in the diagram above.

Use an algorithm included in a presentation 19 Materialized views to find no more than two other views that can be materialized in order to reduce the costs of view processing in the best way.

Question 5 (7 marks)

Consider a fragment of simple JDBC application listed below. It is a typical example of a pretty poor, from performance point of view, JDBC program. Rewrite a code written below to improve the performance of the application it is included in. There is no need to write the entire JDBC application.

Explain all details why the original application takes long time to provide the results and why your version of JDBC code is more efficient than the original one.

```
try{
      Statement stmt = conn.createStatement();
      ResultSet rset = stmt.executeQuery( "SELECT * FROM LINEITEM" );
      int counter = 0;
      float total; = 0.0;
      while ( rset.next() )
        {
            total = total + rset.getFloat(4);
            counter++;
      }
      System.out.println( "Result: " + total/counter );
    }
```

Question 6 (10 marks)

Consider the `SELECT` statements given below. Each one of the given `SELECT` statements joins two or more relational tables. For each `SELECT` statement propose the best method for the implementation of the join algorithm. **Justify your choice ! Note, that answers without the exhaustive and correct justifications score no marks!**

Consider the following implementations of join operation:

(i) Cartesian product join
(ii) Nested loop join
(iii) Nested loop join with one or both arguments kept in transient memory
(iv) Index-based join
(v) Sort-merge join
(vi) Hash join
(vii) Hash antijoin

Assume that no more than 50 Mbytes of transient memory can be invested into the computations of join operation and that size of a bucket in hash implementation of join operation is always less than 5 Mbytes.

The sizes of relevant relational tables are listed at the bottom of the **Introduction** page of the final examination paper.

A solution of each one of the cases listed below is worth 2 marks.

(1)
```
SELECT *
FROM ORDERS, LINEITEM
WHERE ON ORDERS.O_ORDERKEY = LINEITEM.L_ORDERKEY;
```
(2 marks)

(2)
```
SELECT *
FROM PART JOIN PARTSUPP
          ON PART.P_AMOUNT = PARTSUPP.PS_AVAILQTY;
```
(2 marks)

(3)
```
SELECT *
FROM PART
WHERE EXISTS (SELECT *
              FROM LINEITEM
              WHERE PART.P_PARTKEY = LINEITEM.L_PARTKEY);
```
(2 marks)

(4)
```
SELECT *
FROM ORDERS
WHERE NOT EXISTS (  SELECT *
                    FROM LINEITEM
                    WHERE ORDERS.O_ORDERKEY = LINEITEM.L_ORDERKEY  );
```
(2 marks)

(5)
```
SELECT *
FROM LINEITEM
WHERE LINEITEM.L_QUANTITY > (   SELECT L_QUANTITY
                                FROM LINEITEM
                                WHERE L_ORDERKEY = 1 AND L_LINENUMBER = 1  );
```
(2 marks)

Question 7 (8 marks)

Consider three database transactions given below.

Transaction 1

```
UPDATE PART
SET P_SIZE = P_SIZE + 1 WHERE P_PARTKEY = 1;
UPDATE PART
SET P_SIZE = P_SIZE + 2 WHERE P_PARTKEY = 2;
UPDATE PART
SET P_SIZE = P_SIZE + 3 WHERE P_PARTKEY = 3;

COMMIT;
```

Transaction 2

```
UPDATE PART
SET P_SIZE = P_SIZE + 3 WHERE P_PARTKEY = 5;

UPDATE PART
SET P_SIZE = P_SIZE + 3 WHERE P_PARTKEY = 3;

UPDATE PART
SET P_SIZE = P_SIZE + 2 WHERE P_PARTKEY = 2;

COMMIT;
```

Transaction 3

```
UPDATE PART
SET P_SIZE = P_SIZE + 4 WHERE P_PARTKEY = 4;

UPDATE PART
SET P_SIZE = P_SIZE + 23 WHERE P_PARTKEY IN (2,3);

COMMIT;
```

Use a technique of SC-graphs to "chop" the transactions into smaller transactions such that their concurrent processing is more efficient.

Draw SC-graph and rewrite the transaction with inserted COMMIT statements that "chop" the transactions into the smaller pieces.

End of Examination