Family Name	....
First Name	....

SCIT

School of Computing and

Information Technology

CSCI317
Database Performance Tuning

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

S3-2020 FINAL EXAMINATION

Date: ???

Time: ???

Exam value: **40% of the subject assessment**

Marks available: **40 marks**

DIRECTIONS TO CANDIDATES

(1) The answers to the questions included in the final examination must be hand written with a BLACK or DARK BLUE PEN on the WHITE PIECES of paper in A4 format. No pencil and no other colour of paper is allowed.

(2) When finished, take the pictures of the hand-written solution, save the pictures in files (`jpeg`, `jpg`, `gif`, `bmp`, `png` formats are all acceptable), and submit the files through Moodle. Using mobile phone cameras is all right. It is possible to take more than one picture per answer to assure the good readability of an answer. The marks will be deducted for submissions in the different formats. No more than 20 files can be submitted and no more than 200Mbytes can be submitted. Please well plan your pictures.

(3) The file must have the names indicating a number of the respective question in the final examination paper like `q1`, `q2`, ... and `q1-1`, `q1-2`, ... when more than one picture is used for an answer of a question. Marks will be deducted for the incorrect file names.

(4) All answers including the drawings must be hand written. No printed material will be evaluated.

(5) Marks will be deducted for the late submissions at a rate of 1 mark per 1 minute late.

Copyright SCIT, University of Wollongong, 2020

Introduction

The questions 2, 4, 5, and 6 of the examination paper are related to the following simplified version of TPC-H 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,
C_ACCTBAL       NUMBER(6)       NOT NULL,
C_PHONE         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,
L_TAX           NUMBER(4,2)     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) );
```

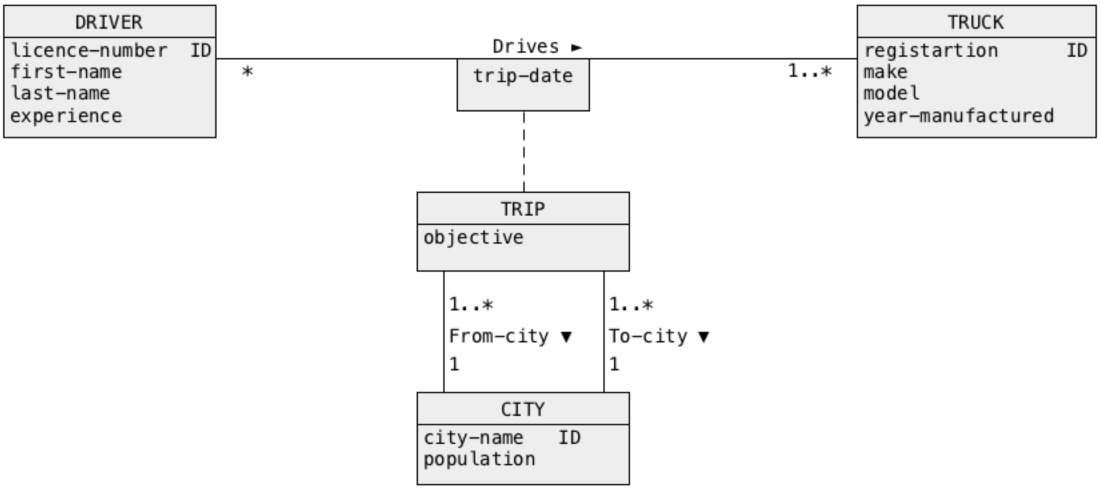
Assume that, the relational tables listed above occupy the following amounts of disk storage:

CUSTOMER	100 Mbytes
PART	40 Mbytes
PARTSUPP	100 Mbytes
ORDERS	200 Mbytes
LINEITEM	700 Mbytes

Copyright SCIT, University of Wollongong, 2020

Question 1 (7 marks)

The following conceptual schema represents a database domain where the drivers use the trucks for the trips from city to city. We assume that a driver can make at most one trip per day. Each trip has an objective, like for example delivery of the ordered items, collection of parcels to be delivered to another place, etc. All other attributes are self-explanatory.



(1) Perform simplification of the conceptual schema above and re-draw the simplified conceptual schema.

(2 marks)

(2) We would like to improve the performance of the following class of applications:

Find the first and the last names of drivers (attributes `first-name`, `last-name` in a class `DRIVER`) who travelled between two given cities (attribute `city-name` in a class `CITY`) and used a vehicle manufactured before a given date (attribute `year-manufactured` in a class `TRUCK`).

The following application belongs to the class of applications given above.

Find the first and the last names of drivers who travelled from `Dapto` to `Sydney` and used an old vehicle manufactured before a year `2000`.

Find the denormalizations of the simplified conceptual schema that improves the performance of the class of applications described above. Re-draw the simplified conceptual schema after the denormalizations.

(5 marks)

Copyright SCIT, University of Wollongong, 2020

Question 2 (7 marks)

Consider the following fragment of query processing plan.

```
------------------------------------------------------------------------------------------
| Id  | Operation              | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |          |   317K|    39M|       | 17968   (1)| 00:00:01 |
|*  1 |  HASH JOIN             |          |   317K|    39M|       | 17968   (1)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL    | CUSTOMER | 40091 |   430K|       |   390   (1)| 00:00:01 |
|*  3 |   HASH JOIN RIGHT ANTI|          |   317K|    35M|  3080K| 17577   (1)| 00:00:01 |
|*  4 |    TABLE ACCESS FULL   | LINEITEM |   150K|  1318K|       | 12153   (1)| 00:00:01 |
|*  5 |    TABLE ACCESS FULL   | ORDERS   |   450K|    46M|       |  2698   (1)| 00:00:01 |
------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("O_CUSTKEY"="C_CUSTKEY")
   2 - filter("C_ACCTBAL">200)
   3 - access("O_ORDERKEY"="L_ORDERKEY")
   4 - filter("L_TAX">0.1)
   5 - filter("O_CUSTKEY">=0)
```

(1) Find and draw a syntax tree of the query processing plan listed above. To draw a syntax tree, use the relational algebra operations (and NOT Oracle query processing plan operations) explained during the lecture classes.

(3 marks)

(2) Discover and write `SELECT` statement that may have a query processing plan listed above.

(4 marks)

Copyright SCIT, University of Wollongong, 2020

Question 3 (6 marks)

A relational table PARTSUPP contains information about the part supplied by suppliers.

PARTSUPP(supplier#, part#, quantity, shipdate)

A relational table PARTSUPP has a composite primary key (supplier#, part#, shipdate)

Assume that:

(i) a relational table PARTSUPP occupies 5000 data blocks,
(ii) a blocking factor in a relational table PARTSUP is 100 rows per block,
(iii) a relational table PARTSUPP contains information about 100 suppliers,
(iv) a relational table PARTSUPP contains information about 500 parts,
(v) a primary key is automatically indexed,
(vi) an attribute part# is indexed,
(vii) all indexes are implemented as B*-trees with a fanout equal to 20,
(viii) a leaf level of an index on attribute part# consists of 50 data blocks,
(ix) a leaf level of an index on primary key consists of 700 data blocks.

For each one of the following queries briefly describe how the database system processes each query and estimate the total number of read block operations needed to compute each query.

(1)
```
SELECT quantity
FROM PARTSUPP
WHERE supplier# = 7 AND part# = 1 AND shipdate ='01-DEC-2019';
```

(2)
```
SELECT quantity
FROM PARTSUP
WHERE part# = 100 OR shipdate > '01-JAN-2020';
```

(3)
```
SELECT part#, COUNT(*)
FROM PARTSUPP
GROUP BY part#;
```

(4)
```
SELECT supplier#, part#, quantity
FROM PARTSUPP
ORDER BY supplier#, part#;
```

(5)
```
SELECT COUNT(*)
FROM PARTSUP
WHERE quantity > 1000 AND shipdate > '01-JAN-2020';
```

(6)
```
SELECT *
FROM PARTSUP
WHERE part# = 12345;
```

Copyright SCIT, University of Wollongong, 2020

Question 4 (6 marks)

Consider the following SELECT statements.

(1)
```
SELECT C_NATIONKEY, COUNT(*)
FROM CUSTOMER
GROUP BY C_NATIONKEY;
```

(2)
```
SELECT C_NATIONKEY, C_ACCTBAL
FROM CUSTOMER
ORDER BY C_NATIONKEY, C_ACCTBAL
```

(3)
```
SELECT COUNT(C_PHONE)
FROM CUSTOMER;
```

(4)
```
SELECT C_NATIONKEY, SUM(C_ACCTBAL)
FROM CUSTOMER
GROUP BY C_NATIONKEY;
```

(5)
```
SELECT *
FROM CUSTOMER
WHERE C_NATIONKEY = 12345 AND C_NAME = 'JAMES'
```

(6)
```
SELECT C_NAME
FROM CUSTOMER
WHERE C_ACCTBAL =100;
```

(1) Find the smallest number of indexes that improve performance of all queries listed above.
(3 marks)

(2) For each query briefly explain how the indexes will be used to process a query.
(3 marks)

Copyright SCIT, University of Wollongong, 2020

Question 5 (6 marks)

Consider the following SELECT statements.

(1)
```
SELECT C_CUSTKEY
FROM CUSTOMER
WHERE ( SELECT COUNT(*)
        FROM ORDERS
        WHERE ORDERS.O_CUSTKEY = CUSTOMER.C_CUSTKEY ) > 10;
```

(2)
```
SELECT DISTINCT (SELECT COUNT(*)
                 FROM PART P
                 WHERE P.P_BRAND = PART.P_BRAND) TOTAL, P_BRAND
FROM PART;
```

(3)
```
CREATE INDEX IDX ON PART(P_NAME);

SELECT *
FROM PART
WHERE (UPPER(P_NAME) = 'BOLT' AND P_RETAILPRICE > 2) ;

DROP INDEX IDX;
```

(4)
```
SELECT O_ORDERKEY, O_CUSTKEY
FROM ORDERS
WHERE O_TOTALPRICE > 10
 UNION
SELECT O_ORDERKEY, O_CUSTKEY
FROM ORDERS
WHERE O_TOTALPRICE < 5;
```

Find and write more efficient implementations of SELECT statements listed above.

Copyright SCIT, University of Wollongong, 2020

Question 6 (8 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 your version of JDBC code is more efficient than the original one.

```
ResultSet rset1 = stmt1.executeQuery(
                  "SELECT P_PARTKEY FROM PART ORDER BY P_NAME" );
long p_partkey = 0;
while ( rset1.next() )
{
  p_partkey = rset1.getInt(1);
  ResultSet rset2 = stmt2.executeQuery(
                    "SELECT COUNT(*) FROM LINEITEM " +
                    "WHERE L_PARTKEY = " + p_partkey );
  long total;
  while ( rset2.next() )
  {
    total = rset2.getInt(1);
    if (total >= 30 )
      System.out.println( p_partkey + "   " + total);
  }
}
```

End of Examination

Copyright SCIT, University of Wollongong, 2020