School of Computing and Information Technology

Student to complete:

Family name	
Other names	
Student number	
Table number	

CSCI317
Database Performance Tuning

Final Examination Paper
Session 3 2022

Exam duration	3 hours
Weighting	40 % of the subject assessment
Marks available	40 marks
Items permitted by examiner	Non-programmable calculator
Directions to students	7 questions to be answered. Marks for each question are shown beside the question. All answers must be written in the answer booklet provided.

This exam paper must not be removed from the exam venue

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.

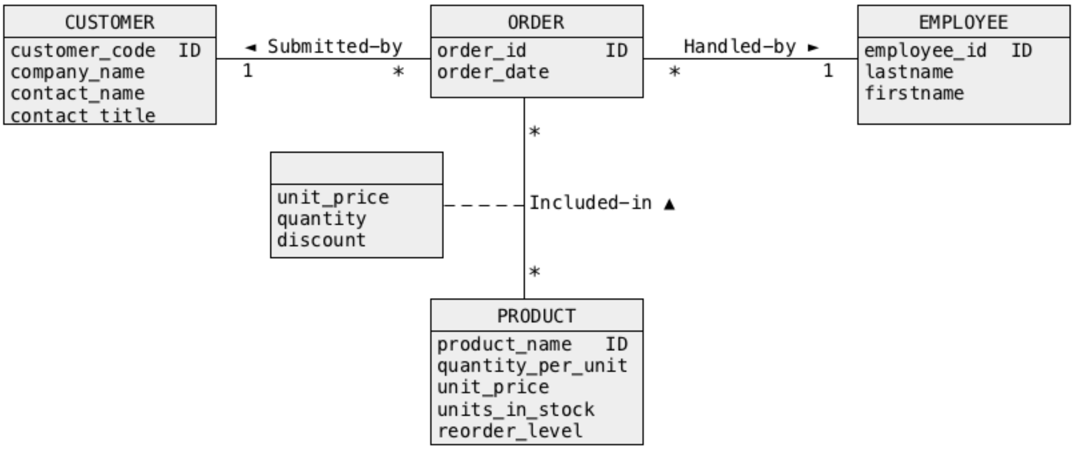
```
CUSTOMER(  C_CUSTKEY        NUMBER(12)      NOT NULL,
           C_NAME           VARCHAR(25)     NOT NULL,
           C_ADDRESS        VARCHAR(40)     NOT NULL,
           C_NATIONKEY      NUMBER(12)      NOT NULL,
      CONSTRAINT CUSTOMER_PKEY PRIMARY KEY(C_CUSTKEY) );

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 (5 marks)

The following conceptual schema represents a database domain where customers submit orders handled by employees and orders include products.



An objective of this task is to apply the denormalization of classes of objects to improve the performance of the following class of applications.

> *Find the full names of employees who handled at least one order submitted by a company (attribute `company-name` in a class `CUSTOMER`) and such that an order included a product with a given name (attribute `product-name` in a class `PRODUCT`) and with a given quantity (attribute `quantity`).*

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

> *Find the full names of employees who handled at least one order submitted by a company `Golden Bolts` and such that an order included `bolts` with a quantity higher than `1000`.*

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

(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.
(2 marks)

(3) To further improve performance, apply appropriate decompositions and redraw the final conceptual schema.
(2 marks)

Question 2 (5 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` statement to create the indexes.

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

(2) 1 mark
```
SELECT AVG(L_QUANTITY)
FROM ORDERS JOIN LINEITEM
             ON O_ORDERKEY = L_ORDERKEY;
```

(3) 1 mark
```
SELECT AVG(OPS_AVAILQTY)
FROM PARTSUPP
WHERE PS_SUPPNAME = 'James';
```

(4) 1 mark
```
SELECT P_NAME,
FROM PART
WHERE P_BRAND = 'RUBBISH'
ORDER BY P_NAME;
```

(5) 1 mark
```
SELECT C_NAME
FROM CUSTOMER
WHERE C_NATIONKEY = 'SG';
MINUS
SELECT C_NAME
FROM CUSTOMER
WHERE C_ADDRESS LIKE '%Bugis%';
```

Question 3 (6 marks)

Assume that a relational table `ORDERS` contains information about the orders submitted by the customers.

```
ORDERS(order#, order_date, product, quantity, price_per_unit)
```

A relational table `ORDERS` has a primary key `(order#)`.

Assume that:

(i) a relational table `ORDERS` occupies 1000 data blocks,
(ii) a blocking factor in a relational table `ORDERS` is 50 rows per block,
(iii) a relational table `ORDERS` contains information about 200 products,
(iv) a relational table `ORDERS` contains information about 100 prices per unit,
(v) a primary key is automatically indexed,
(vi) an attribute `product` is indexed,
(vii) all indexes are implemented as B*-trees with a fanout equal to 10,
(viii) a leaf level of an index on an attribute `product` consists of 30 data blocks,
(ix) a leaf level of an index on primary key consists of 200 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. There is no need to perform the final computations. A correctly constructed formula filled with the appropriate constants is completely sufficient.

(1) 1 mark
```
SELECT product
FROM ORDERS
WHERE product = 'bolt' OR quantity = 100;
```

(2) 1 mark
```
SELECT count(*)
FROM ORDERS
WHERE product IN ('bolt', 'screw');
```

(3) 1 mark
```
SELECT product, COUNT(*)
FROM ORDERS
GROUP BY product
HAVING count(*) > 5;
```

(4) 1 mark
```
SELECT order#, product, quality
FROM ORDERS
ORDER BY order#, product;
```

(5) 1 mark
```
SELECT *
FROM ORDERS
WHERE order# = 12345 AND product = 'bolt';
```

(6) 1 mark
```
SELECT product
FROM ORDERS;
```

Question 4 (6 marks)

Consider `SELECT` statements listed below.

```
SELECT L_PARTKEY, L_TAX, COUNT(*)
FROM LINEITEM
GROUP BY L_PARTKEY,L_TAX;

SELECT O_TOTALPRICE, COUNT(*)
FROM ORDERS
GROUP BY O_TOTALPRICE;

SELECT L_TAX, L_QUANTITY, COUNT(*)
FROM LINEITEM
GROUP BY L_TAX, L_QUANTITY;

SELECT O_CLERK, O_TOTALPRICE, COUNT(*)
FROM ORDERS
GROUP BY O_CLERK, O_TOTALPRICE;
```

Write SQL statements that create the smallest number of materialized views that can be automatically used to speed up the processing of `SELECT` statements given above.

Question 5 (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.

(1) 5 marks
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.

```
Statement stmt1 = conn.createStatement();
Statement stmt2 = conn.createStatement();
ResultSet rset1 = stmt1.executeQuery(
                                "SELECT DISTINCT C_CUSTKEY " +
                                "FROM CUSTOMER" );
long c_custkey = 0;
while ( rset1.next() )
{
  c_custkey = rset1.getInt(1);
  ResultSet rset2 = stmt2.executeQuery( "SELECT * " +
                                        "FROM ORDERS " +
                                        "WHERE C_CUSTKEY = " + c_custkey );
  long o_orderkey = 0;
  long total = 0;
  while ( rset2.next() )
  {
    o_orderkey = rset2.getInt(1);
    total++;
  }
  System.out.println( c_custkey + "   " + counter);
}
```

(2) 3 marks
Explain all details why your version of JDBC code is more efficient than the original one.

Question 6 (4 marks)

Consider `SELECT` statements listed below. Rewrite each one of the statements listed below into an equivalent `SELECT` statement and such that its processing will take less read block operations then the processing of the original one.

(1) 1 mark

```
SELECT L_ORDERKEY, L_PARTKEY
FROM ORDERS JOIN LINEITEM
     ON ORDERS.O_ORDERKEY = LINEITEM.O_ORDERKEY;
```

(2) 1 mark

```
SELECT *
FROM PART
WHERE P_PARTKEY = 101
UNION
SELECT *
FROM PART
WHERE P_PARTKEY = 102;
```

(3) 1 mark

```
SELECT C_NAME, C_ADDRESS
FROM CUSTOMER
WHERE C_CUSTKEY IN ( SELECT C_CUSTKEY
                         FROM CUSTOMER
                         WHERE C_NAME = 'Jones');
```

(4) 1 mark

```
SELECT O_CUSTKEY
FROM ORDERS ORD
WHERE 5 < ( SELECT COUNT(*)
            FROM ORDERS
            WHERE ORDERS.C_CUSTKEY = ORD.C_CUSTKEY );
```

Question 7 (6 marks)

Consider SELECT statement given below.

```
SELECT P_PARTKEY, P_NAME, COUNT(*)
FROM   PART JOIN PARTSUPP
       ON PART.P_PARTKEY = PARTSUPP.PS_PARTKEY
GROUP BY P_PARTKEY, P_NAME
UNION
SELECT P_PARTKEY, P_NAME, 0
FROM   PART
WHERE  P_PARTKEY NOT IN ( SELECT PS_PARTKEY
                          FROM   PARTSUPP );
```

A fragment of query execution plan that describe the extended relational algebra expression for SELECT statement above is the following.

```
-----------------------------------------------------
| Id  | Operation                  | Name          | …
-----------------------------------------------------
|   0 | SELECT STATEMENT           |               |
|   1 |  SORT UNIQUE               |               |
|   2 |   UNION-ALL                |               |
|   3 |    HASH GROUP BY           |               |
|*  4 |     HASH JOIN              |               |
|   5 |      TABLE ACCESS FULL     | PART          |
|   6 |      INDEX FAST FULL SCAN| PARTSUPP_PKEY |
|*  7 |    HASH JOIN ANTI          |               |
|   8 |     TABLE ACCESS FULL      | PART          |
|   9 |     INDEX FAST FULL SCAN | PARTSUPP_PKEY |
-----------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("PART"."P_PARTKEY"="PARTSUPP"."PS_PARTKEY")
   7 - access("P_PARTKEY"="PS_PARTKEY")
```

(1) Draw a syntax tree of a query processing plan given above.

(3 marks)

(2) A query processing plan given above reveals that relational table PART and an index PARTSUPP_PKEY on a primary key of PARTSUPP table are accessed twice (see lines 5, 6 and 8, 9 above). Find and write SELECT statement that retrieves the same information from the relational tables PART and PARTSUPP and such that it accesses a relational table PART only once.

(3 marks)

End of Examination