Assignment 1 (10% of total marks)

Due date: Sunday, 24 October, 2021 by 9:00 pm SGT.

Scope:

The tasks of this assignment cover **functional dependency and indexing**. The assignment covers the topics discussed in lecture 1, 2, and 3.

Assessment criteria:

Marks will be awarded for:

- Correct,
- Comprehensive, and
- Appropriate

application of the materials covered in this subject.

Assignment Specification:

Task 1 (3.0 marks)
Analysis of relational schemas and normalization

A book is described by a catalog number, a book title, author's first name, author's last name, a publisher name, the year the book was published, and a price of the book. An author may write many books and each book may be written by one or more authors. The information is stored in a relational table book as follow:

BOOK (CatalogNum, BookTitle, AuthorFName, AuthorLName, PublisherName, YearOfPublication, Price)

The following dependencies exist in the relational table BOOK:

- $CatalogNum \rightarrow BookTitle, PublisherName, YearOfPublication$
- $BookTitle, PublisherName, YearOfPublication \rightarrow Price$
- $AuthorFirstName, AuthorLastName, BookTitle \rightarrow CatalogNum$

(i) Find all the minimal super key of the relational table BOOK. List the derivations of all minimal keys.
(ii) Identify the highest normal form of the relational table BOOK. List the justifications for each highest normal form found.
(iii) Decompose the relational table BOOK into minimal number of relational tables in BCNF. List all relational tables obtained from the decompositions.

Copyright SCIT, University of Wollongong, 2021

Sample solution includes but is not limited to the following:

(i) Find all the minimal super key of the relational table BOOK. List the derivations of all minimal keys.

- If $CatalogNum \rightarrow BookTitle, PublisherName, YearOfPublication$ and $BookTitle, PublisherName, YearOfPublication \rightarrow Price$, then through transitivity axiom $CatalogNum \rightarrow Price$.
- Since an author may write many books and a book may be written by many authors, through augmentation axiom, $CatalogNum, AuthorFirstName, AuthorLastName \rightarrow BookTitle, PublisherName, YearOfPublication, Price, AuthorFirstName, AuthorLastName$.
- If $CatalogNum, AuthorFirstName, AuthorLastName \rightarrow BookTitle, PublisherName, YearOfPublication, Price, AuthorFirstName, AuthorLastName$ is valid in the relational table BOOK, and it covers the entire relational schema, then the left-hand-side of the functional dependency $(\boldsymbol{CatalogNum, AuthorFirstName, AuthorLastName})$ is a minimal super key.
- If $AuthorFirstName, AuthorLastName, BookTitle \rightarrow CatalogNum$, and $CatalogNum \rightarrow BookTitle, PublisherName, YearOfPublication$, then through transitivity axiom, $AuthorFirstName, AuthorLastName, BookTitle \rightarrow CatalogNum, BookTitle, PublisherName, YearOfPublication$.
- Similarly, if $BookTitle, PublisherName, YearOfPublication \rightarrow Price$, then through transitivity axiom, $AuthorFirstName, AuthorLastName, BookTitle \rightarrow CatalogNum, BookTitle, PublisheName, YearOfPublication, Price$.
- If $AuthorFirstName, AuthorLastName, BookTitle \rightarrow CatalogNum, BookTitle, PublisherName, YearOfPublication, Price$ is valid in the relational table BOOK, and it covers the entire relational schema, then the left-hand-side of the functional dependency $(\boldsymbol{AuthorFirstName, AuthorLastName, BookTitle})$ is a minimal super key.

(ii) Identify the highest normal form of the relational table BOOK. List the justifications for each highest normal form found.

- Since in the functional dependency $CatalogNum, AuthorFirstName, AuthorLastName \rightarrow BookTitle, PublisherName, YearOfPublication, Price, AuthorFirstName, AuthorLastName$, there exist a partial dependency $CatalogNum \rightarrow BookTitle, PublisherName, YearOfPublication$. The relational table BOOK has a partial dependency violation. Hence, the highest normal form of the relational table BOOK is in First Normal Form (1NF).

Copyright SCIT, University of Wollongong, 2021

- If $AuthorFirstName, AuthorLastName, BookTitle \rightarrow CatalogNum, BookTitle, PublisherName, YearOfPublication, Price$ is considered, then there a transitive dependency $AuthorFirstName, AuthorLastName, BookTitle \rightarrow CatalogNum$ and $CatalogNum \rightarrow PublisherName, YearOfPublication$. This makes the relational table BOOK not qualified to be in 3NF. In addition, there exist a non-trivial functional dependency $CatalogNum \rightarrow BookTitle$. This further make the relational table BOOK to violate the requirement to be in BCNF. Hence, the highest normal form of the relational table BOOK is in Second Normal Form (2NF).

(iii) Decompose the relational table BOOK into minimal number of relational tables in BCNF. List all relational tables obtained from the decompositions.

BDE-> CFGHDE

In the first scenario, $CatalogNum, AuthorFirstName, AuthorLastName \rightarrow BookTitle, PublisherName, YearOfPublication, Price, AuthorFirstName, AuthorLastName$, to normalize the relational table BOOK into BCNF, we first remove the partial dependency from the relational table BOOK, and decompose the relational table into:

$BOOKCATALOG2NF(CatalogNum, BookTitle, PublisherName, YearOfPublication, Price)$
$PK: CatalogNum$

$AUTHORCATALOG2NF(CatalogNum, AuthorFirstName, AuthorLastName)$
$PK: (CatalogNum, AuthorFirstName, AuthorLastName)$
$FK: (CatalogNum)\ references\ BOOKCATALOG2NF(CatalogNum)$

Next, bring the relational table BOOKCATALOG2NF to 3NF by removing the transitive dependency $BookTitle, PublisherName, YearOfPublicatiion \rightarrow Price$, and we have the following:

$BOOKPRICE3NF(BookTitle, PublisherName, YearOfPublication, Price)$
$PK: (BookTitle, PublisherName, YearOfPublication)$

$BOOKCATALOG3NF(CatalogNum, BookTitle, PublisherName, YearOfPublication)$
$PK: CatalogNum$
$FK: (BookTitle, PublisherName, YearOfPublication)\ references$
$BOOKPRICE3NF(BookTitle, PublisherName, YearOfPublication)$

Since the relational table AUTHORCATALOG2NF has no transitive dependency, it is qualified to be in 3NF. Hence,

$AUTHORCATALOG3NF(CatalogNum, AuthorFirstName, AuthorLastName)$
$PK: (CatalogNum, AuthorFirstName, AuthorLastName)$
$FK: (CatalogNum)\ references\ BOOKCATALOG3NF(CatalogNum)$

Copyright SCIT, University of Wollongong, 2021

Since the 3 relational tables do not have non-trivial functional dependency, the 3 relational table are also in BCNF.

The final relational tables in BCNF are:

$BOOKPRICEBCNF(BookTitle, PublisherName, YearOfPublication, Price)$
$PK: (BookTitle, PublisherName, YearOfPublication)$

$BOOKCATALOGBCNF(CatalogNum, BookTitle, PublisherName, YearOfPublication)$
$PK: CatalogNum$
$FK: (BookTitle, PublisherName, YearOfPublication)\ references$
$BOOKPRICEBCNF(BookTitle, PublisherName, YearOfPublication)$

$AUTHORCATALOGBCNF(CatalogNum, AuthorFirstName, AuthorLastName)$
$PK: (CatalogNum, AuthorFirstName, AuthorLastName)$
$FK: (CatalogNum)\ references\ BOOKCATALOGBCNF(CatalogNum)$

Deliverables
A file solution1.pdf with the outcomes of the steps (i), (ii), and (iii) listed above. Note, that "educated guesses" of the solutions score no marks. You must provide the complete justifications of your answers.

Submission of a file with a different name and/or different extension and/or different type scores no marks!

Copyright SCIT, University of Wollongong, 2021

Task 2 (4.0 marks)
Indexing
Using the relational table LINEITEM of the sample database TPCHR, for each one of the queries listed below:

i. Find all the discount (l_discount) of all the items that are shipped (l_shipdate) most recently. Hint. Most recently mean the latest shipment date.

ii. Find the total number of items shipped by air (l_shipmode) in 1998 (l_shipdate).

iii. Find the order number (l_orderkey) and item number (l_linenumber) that have the highest discount (l_discount).

iv. Find the total number of item per line status (l_linestatus). List the line status and the total items per line status.

v. Find the order key (l_orderkey), line item number (l_linenumber), line status (l_linestatus), shipment date (l_shipdate) and shipment mode (l_shipmode) of all orders with the order number (l_orderkey) 1795718, 1799046, and 1794626.

a) Construct an SQL statement that produces the required output specified in the statement. **(1.5 marks)**

b) Find the best possible indexing of a relational table LINEITEM. The best possible indexing means that a database system will compute a query with an index proposed by you using the smallest number of read block operations. Note that you can create only one index per query, and there is no need to find indexes that speed up the processing of more than one query. Use the explain plan and show plan statements to justify your solutions. **(2.5 marks)**

Deliverables

A file solution3.pdf with CREATE INDEX statements that improve the performance of the queries listed (i, ii, iii, iv, and v above) and the execution plan generated.

Please remember that you must consider each one of the queries as an individual case! Please remember that all relational tables are large enough to make full table scans more time consuming that accessing the tables through an index! It means that any solution in which an index is not used for query processing is incorrect.

```
SQL> --
SQL> set echo on
SQL> set feedback on
SQL> set linesize 100
```


Copyright SCIT, University of Wollongong, 2021

```
SQL> set pagesize 1000
SQL> --
SQL> /* Task 2(i)
SQL> Find all the discount (l_discount) that have the most recent shipment date
(l_shipdate).
SQL> List the discount in descending order
SQL> */
SQL> --
SQL> explain plan for
  2  select l_discount
  3  from lineitem
  4  where to_char(l_shipdate,'yyyymmdd') in (
  5  select max(to_char(l_shipdate,'yyyymmdd'))
  6  from lineitem)
  7  order by l_discount desc;

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
------------------
Plan hash value: 2368494269

--------------------------------------------------------------------------------
-------
| Id  | Operation            | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)|
Time     |
--------------------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT     |          | 14000 |   300K|       | 24415   (1)|
00:00:01 |
|   1 |  SORT ORDER BY       |          | 14000 |   300K|   448K| 24415   (1)|
00:00:01 |
|*  2 |   TABLE ACCESS FULL  | LINEITEM | 14000 |   300K|       | 12170   (1)|
00:00:01 |
|   3 |    SORT AGGREGATE    |          |     1 |     9 |       |            |
|
|   4 |     TABLE ACCESS FULL| LINEITEM |  1400K|    12M|       | 12149   (1)|
00:00:01 |
--------------------------------------------------------------------------------
-------

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

   2 - filter(TO_CHAR(INTERNAL_FUNCTION("L_SHIPDATE"),'yyyymmdd')= (SELECT
              MAX(TO_CHAR(INTERNAL_FUNCTION("L_SHIPDATE"),'yyyymmdd')) FROM
"LINEITEM"
              "LINEITEM"))

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

22 rows selected.

SQL> --
SQL> /*
SQL> Create an index on l_shipdate to improve the retrieval process of the query.
```


Copyright SCIT, University of Wollongong, 2021

```
SQL> */
SQL> create index A1Task2iIdx on lineitem(to_char(l_shipdate,'yyyymmdd'));

Index created.

SQL> --
SQL> explain plan for
  2  select l_discount
  3  from lineitem
  4  where to_char(l_shipdate,'yyyymmdd') in (
  5  select max(to_char(l_shipdate,'yyyymmdd'))
  6  from lineitem)
  7  order by l_discount desc;

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
------------------
Plan hash value: 4098712802

--------------------------------------------------------------------------------
------------------
| Id  | Operation                           | Name        | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------------------
------------------
|   0 | SELECT STATEMENT                    |             |     1 |    19 |  7024
(1)| 00:00:01 |
|   1 |  SORT ORDER BY                      |             |     1 |    19 |  7024
(1)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM    |     1 |    19 |  7020
(1)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN                 | A1TASK2IIDX |  5600 |       |     3
(0)| 00:00:01 |
|   4 |     SORT AGGREGATE                  |             |     1 |     6 |
|           |
|   5 |      INDEX FULL SCAN (MIN/MAX)      | A1TASK2IIDX |     1 |     6 |     3
(0)| 00:00:01 |
--------------------------------------------------------------------------------
------------------

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

   3 - access(TO_CHAR(INTERNAL_FUNCTION("L_SHIPDATE"),'yyyymmdd')= (SELECT
              MAX(TO_CHAR(INTERNAL_FUNCTION("L_SHIPDATE"),'yyyymmdd')) FROM
"LINEITEM" "LINEITEM"))

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

22 rows selected.

SQL> --
SQL> /* Task 2(ii)
SQL> Find the total number of items shipped by air (l_shipmode) in 1998
(l_shipdate).
```


Copyright SCIT, University of Wollongong, 2021

```
SQL> */
SQL> explain plan for
  2  select count(*)
  3  from lineitem
  4  where upper(l_Shipmode) = 'AIR'
  5  and to_char(l_shipdate,'yyyy') = '1998';

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
------------------
Plan hash value: 2287326370

-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |     1 |    21 | 12166   (1)| 00:00:01 |
|   1 |  SORT AGGREGATE    |          |     1 |    21 |            |          |
|*  2 |   TABLE ACCESS FULL| LINEITEM | 16942 |   347K| 12166   (1)| 00:00:01 |
-------------------------------------------------------------------------------

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

   2 - filter(UPPER("L_SHIPMODE")='AIR' AND
              TO_CHAR(INTERNAL_FUNCTION("L_SHIPDATE"),'yyyy')='1998')

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

19 rows selected.

SQL> --
SQL> -- Create an index on shipment year to improve the query performance.
SQL> create index A1Task2iiIdx on lineitem(l_shipmode,
to_char(l_shipdate,'yyyy'));

Index created.

SQL> --
SQL> explain plan for
  2  select count(*)
  3  from lineitem
  4  where upper(l_Shipmode) = 'AIR'
  5  and to_char(l_shipdate,'yyyy') = '1998';

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
------------------
Plan hash value: 4145958068

--------------------------------------------------------------------------------
```


Copyright SCIT, University of Wollongong, 2021

```
----
| Id  | Operation             | Name         | Rows  | Bytes | Cost (%CPU)| Time
|
-------------------------------------------------------------------------------
----
|   0 | SELECT STATEMENT      |              |     1 |    16 |  2626   (1)|
00:00:01 |
|   1 |  SORT AGGREGATE       |              |     1 |    16 |            |
|
|*  2 |   INDEX FAST FULL SCAN| A1TASK2IIIDX | 16942 |   264K|  2626   (1)|
00:00:01 |
-------------------------------------------------------------------------------
----

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

   2 - filter(UPPER("L_SHIPMODE")='AIR' AND
              TO_CHAR(INTERNAL_FUNCTION("L_SHIPDATE"),'yyyy')='1998')

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

19 rows selected.

SQL> --
SQL> /* Task 2(iii)
SQL> Find the order number (l_orderkey) and item number (l_linenumber) that have
the
SQL> highest discount (l_discount).
SQL> */
SQL> explain plan for
  2  select l_orderkey, l_linenumber
  3  from lineitem
  4  where l_discount = (select max(l_discount) from lineitem);

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
-----------------
Plan hash value: 4120815904

-------------------------------------------------------------------------------
| Id  | Operation           | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |          |     1 |    39 | 24295   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL  | LINEITEM |     1 |    39 | 12148   (1)| 00:00:01 |
|   2 |   SORT AGGREGATE    |          |     1 |    13 |            |          |
|   3 |    TABLE ACCESS FULL| LINEITEM |  1400K|    17M| 12146   (1)| 00:00:01 |
-------------------------------------------------------------------------------

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

   1 - filter("L_DISCOUNT"= (SELECT MAX("L_DISCOUNT") FROM "LINEITEM"
              "LINEITEM"))
```


Copyright SCIT, University of Wollongong, 2021

```
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

20 rows selected.

SQL> --
SQL> -- Create an index on discount to speed up the processing of the query.
SQL> create index A1Task2iiiIdx on lineitem(l_discount);

Index created.

SQL> --
SQL> explain plan for
  2  select l_orderkey, l_linenumber
  3  from lineitem
  4  where l_discount = (select max(l_discount) from lineitem);

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
------------------
Plan hash value: 3457977591

--------------------------------------------------------------------------------
------------------
-

| Id  | Operation                           | Name          | Rows  | Bytes | Cost
(%CPU)| Time
|

--------------------------------------------------------------------------------
------------------
-

|   0 | SELECT STATEMENT                    |               |     1 |    39 |
1739   (0)| 00:00:01
|

|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM      |     1 |    39 |
1736   (0)| 00:00:01
|

|*  2 |   INDEX RANGE SCAN                  | A1TASK2IIIIDX |  5600 |       |
319   (0)| 00:00:01
|

|   3 |    SORT AGGREGATE                   |               |     1 |    13 |
|
|

|   4 |     INDEX FULL SCAN (MIN/MAX)       | A1TASK2IIIIDX |     1 |    13 |
3   (0)| 00:00:01
|

--------------------------------------------------------------------------------
------------------
```


Copyright SCIT, University of Wollongong, 2021

```
-


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

   2 - access("L_DISCOUNT"= (SELECT MAX("L_DISCOUNT") FROM "LINEITEM" "LINEITEM"))

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

20 rows selected.

SQL> --
SQL> /* Task 2(iv)
SQL> Find the total number of item per line status (l_linestatus).
SQL> List the line status and the total items per line status.
SQL> */
SQL> explain plan for
  2  select l_linestatus, count(*)
  3  from lineitem
  4  group by l_linestatus;

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
------------------
Plan hash value: 1773397105

-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |  1400K|  4101K| 12184   (1)| 00:00:01 |
|   1 |  HASH GROUP BY     |          |  1400K|  4101K| 12184   (1)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| LINEITEM |  1400K|  4101K| 12149   (1)| 00:00:01 |
-------------------------------------------------------------------------------

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

13 rows selected.

SQL> --
SQL> -- Create an index on line status (l_linestatus) to speed up the query
processing.
SQL> create index A1Task2ivIdx on lineitem(l_linestatus);

Index created.

SQL> --
SQL> explain plan for
  2  select l_linestatus, count(*)
  3  from lineitem
  4  group by l_linestatus;

Explained.
```


Copyright SCIT, University of Wollongong, 2021

```
SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
------------------
Plan hash value: 1940216922

--------------------------------------------------------------------------------
----
| Id  | Operation             | Name         | Rows  | Bytes | Cost (%CPU)| Time
|
--------------------------------------------------------------------------------
----
|   0 | SELECT STATEMENT      |              |  1400K|  4101K|  1292   (4)|
00:00:01 |
|   1 |  HASH GROUP BY        |              |  1400K|  4101K|  1292   (4)|
00:00:01 |
|   2 |   INDEX FAST FULL SCAN| A1TASK2IVIDX |  1400K|  4101K|  1256   (1)|
00:00:01 |
--------------------------------------------------------------------------------
----

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

13 rows selected.

SQL> --
SQL> /* Task 2(v)
SQL> Find the line status (l_shipmode) of all orders with the number (l_orderkey)
SQL> 1795718, 1799046, and 1794626.
SQL> */
SQL> --
SQL> -- There is no need to create index for this query because the system can use
SQL> -- the primary key index (l_orderkey).
SQL> explain plan for
  2  select l_orderkey, l_linenumber, l_linestatus, l_shipdate, l_shipmode
  3  from lineitem
  4  where l_orderkey in (1795718, 1799046, 1794626);

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
------------------
Plan hash value: 2007773388

--------------------------------------------------------------------------------
------------------
--

| Id  | Operation                           | Name          | Rows  | Bytes |
Cost (%CPU)| Time
 |

--------------------------------------------------------------------------------
```


Copyright SCIT, University of Wollongong, 2021

```
------------------
--

|   0 | SELECT STATEMENT                    |               |   356 | 17800 |
1055   (0)| 00:00:01
 |

|   1 |  INLIST ITERATOR                    |               |       |       |
|
 |

|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM      |   356 | 17800 |
1055   (0)| 00:00:01
 |

|*  3 |    INDEX RANGE SCAN                 | LINEITEM_PKEY |  5600 |       |
19   (0)| 00:00:01
 |

--------------------------------------------------------------------------------
------------------
--


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

   3 - access("L_ORDERKEY"=1794626 OR "L_ORDERKEY"=1795718 OR
"L_ORDERKEY"=1799046)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

19 rows selected.

SQL> --
SQL> -- Drop all the indexes created for this task.
SQL> drop index A1Task2iIdx;

Index dropped.

SQL> drop index A1Task2iiIdx;

Index dropped.

SQL> drop index A1Task2iiiIdx;

Index dropped.

SQL> drop index A1Task2ivIdx;

Index dropped.

SQL> spool off
```

Task 3 (3.0 marks)
Indexing

Copyright SCIT, University of Wollongong, 2021

Using the relational table ORDERS of the sample database TPCHR and the following index created, construct an appropriate SQL statement and satisfies the following requirement:

create index A1Task3Idx on orders(o_orderdate, o_clerk, o_totalprice);

Find SELECT statements that will use the index in the following ways:

a) Execution of the first SELECT statement must traverse the index vertically and it must not access a relational table ORDERS. **(0.6 mark)**

b) Execution of the second SELECT statement must first traverse the index vertically followed by horizontal scan at the leaf level. The retrieval operation must not access a relational table ORDERS. **(0.6 mark)**

c) Execution of the third SELECT statement must traverse the leaf level of the index horizontally and it must not access the relational table ORDERS. **(0.6 mark)**

d) Execution of the fourth SELECT statement must traverse the index vertically and it must access a relational table ORDERS. **(0.6 mark)**

e) Execution of the fifth SELECT statement must first traverse the index vertically followed by horizontal scan at the leaf level. The retrieval operation must access a relational table ORDERS. **(0.6 mark)**

```
SQL> --
SQL> set echo on
SQL> set feedback on
SQL> set linesize 100
SQL> set pagesize 1000
SQL> --
SQL> create index A1Task3Idx on orders(o_orderdate, o_clerk, o_totalprice);

Index created.

SQL> --
SQL> -- Task 3 a
SQL> -- Access the index vertically and must not access the
SQL> -- relational table.
SQL> explain plan for
  2  select count(*)
  3  from orders
  4  where o_orderkey = '123';

Explained.

SQL> --
```


Copyright SCIT, University of Wollongong, 2021

```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-----------------
Plan hash value: 3502495581

-------------------------------------------------------------------------------
| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |             |     1 |    13 |     1   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |             |     1 |    13 |            |          |
|*  2 |   INDEX UNIQUE SCAN| ORDERS_PKEY |     1 |    13 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------

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

   2 - access("O_ORDERKEY"=123)

14 rows selected.

SQL> --
SQL> -- Task 3 b
SQL> -- Access the index vertically first and followed by
SQL> -- horiontal scan at the leaf level.
SQL> explain plan for
  2  select count(*)
  3  from orders
  4  where o_orderdate > '11-Oct-2021';

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-----------------
Plan hash value: 3639741489

-------------------------------------------------------------------------------
| Id  | Operation          | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |            |     1 |     9 |     2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |            |     1 |     9 |            |          |
|*  2 |   INDEX RANGE SCAN| A1TASK3IDX |     1 |     9 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

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

   2 - access("O_ORDERDATE">TO_DATE(' 2021-10-11 00:00:00',
              'syyyy-mm-dd hh24:mi:ss'))

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

19 rows selected.

SQL> --
```


Copyright SCIT, University of Wollongong, 2021

```
SQL> -- Task 3 c
SQL> -- Access the index horizontally and must not access the
SQL> -- relational table.
SQL> explain plan for
  2  select o_orderdate
  3  from orders;

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-----------------
Plan hash value: 341863333

-------------------------------------------------------------------------------
| Id  | Operation            | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |            |   564K|  4962K|   990   (1)| 00:00:01 |
|   1 |  INDEX FAST FULL SCAN| A1TASK3IDX |   564K|  4962K|   990   (1)| 00:00:01 |
-------------------------------------------------------------------------------

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

12 rows selected.

SQL> --
SQL> -- Task 3 d
SQL> -- Access the index vertically and must access the relational
SQL> -- table.
SQL> explain plan for
  2  select *
  3  from orders
  4  where o_orderkey = '123';

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-----------------
Plan hash value: 572458442

-------------------------------------------------------------------------------
--------
| Id  | Operation                   | Name        | Rows  | Bytes | Cost (%CPU)|
Time     |
-------------------------------------------------------------------------------
--------
|   0 | SELECT STATEMENT            |             |     1 |   139 |     2   (0)|
00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| ORDERS      |     1 |   139 |     2   (0)|
00:00:01 |
|*  2 |   INDEX UNIQUE SCAN         | ORDERS_PKEY |     1 |       |     1   (0)|
00:00:01 |
-------------------------------------------------------------------------------
```


Copyright SCIT, University of Wollongong, 2021

```
--------

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

   2 - access("O_ORDERKEY"=123)

14 rows selected.

SQL> --
SQL> -- Task 3 e
SQL> -- Access the index vertically and followed by horizontal
SQL> -- scan of the index at leaf level, and then access the
SQL> -- relational table.
SQL> explain plan for
  2  select *
  3  from orders
  4  where o_orderdate = '11-Oct-2021'
  5  and o_clerk = 'clerk123'
  6  and o_totalprice = 100;

Explained.

SQL> --
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-----------------
Plan hash value: 1914512372

--------------------------------------------------------------------------------
---------------
| Id  | Operation                           | Name       | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------------------
---------------
|   0 | SELECT STATEMENT                    |            |     1 |   139 |     2
(0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| ORDERS     |     1 |   139 |     2
(0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | A1TASK3IDX |     1 |       |     2
(0)| 00:00:01 |
--------------------------------------------------------------------------------
---------------

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

   2 - access("O_ORDERDATE"=TO_DATE(' 2021-10-11 00:00:00', 'syyyy-mm-dd
hh24:mi:ss') AND
              "O_CLERK"='clerk123' AND "O_TOTALPRICE"=100)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

19 rows selected.

SQL> --
SQL> Drop index A1Task3Idx;
```


Copyright SCIT, University of Wollongong, 2021

```
Index dropped.

SQL> spool off
```

Deliverables

A file solution3.pdf with the SELECT statements for each one of the five cases described above.

Submissions

This assignment is due on Sunday, 31 October 2021 by 9:00 pm (21:00 hours) Singapore time.

Submit the files **solution1.pdf**, **solution2.pdf**, and **solutions3.pdf** through Moodle in the following way:

1) Zip all the files (Solution1.pdf, solution2.pdf, and solution3.pdf into one zipped folder.)
2) Access Moodle at **http://moodle.uowplatform.edu.au/**
3) To login use a Login link located in the right upper corner the Web page or in the middle of the bottom of the Web page
4) When successfully logged in, select a site CSCI235 (SP421) Database Systems
5) Scroll down to a section Submissions of Assignments
6) Click at Submit your Assignment 1 here link.
7) Click at a button Add Submission
8) Move the zipped file created in Step 1 above into an area provided in Moodle. You can drag and drop files here to add them. You can also use a link *Add…*
9) Click at a button Save changes,
10) Click at check box to confirm authorship of a submission,
11) When you are satisfied, remember to click at a button Submit assignment.

A policy regarding late submissions is included in the subject outline. Only one submission per student is accepted.

Assignment 1 is an individual assignment and it is expected that all its tasks will be solved individually without any cooperation with the other students. Plagiarism is treated seriously. Students involved will likely receive zero. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or over e-mail.

End of specification

Copyright SCIT, University of Wollongong, 2021