School of Information Technology & Computer Science
University of Wollongong

Session: **3, July 2021**
Lecturer: **Janusz R. Getta**

CSCI317 Database Performance Tuning
Singapore 2021-3
Assignment 1
Published on 3 July 2021

Scope

This assignment includes the tasks related to database administration, implementation of queries in relational algebra, interpretation of query processing plans, discovery of `SELECT` statement from a query processing plan, and optimal distribution of relational table over the persistent storage devices.

This assignment is due by **Saturday, 17 July 2021, 9.00 pm (sharp) Singaporean Time.**

Please read very carefully information listed below.

This assignment contributes to 15% of the total evaluation in the subject.

A submission procedure is explained at the end of specification.

This assignment consists of 5 tasks and specification of each task starts from a new page.

It is recommended to solve the problems before attending a laboratory class in order to efficiently use supervised laboratory time.

A submission marked by Moodle as "late" is treated as a late submission no matter how many seconds it is late.

A policy regarding late submissions is included in the subject outline.

A submission of compressed files (zipped, gzipped, rared, tared, 7-zipped, lhzed, … etc) is not allowed. The compressed files will not be evaluated.

All files left on Moodle in a state "`Draft(not submitted)`" will not be evaluated.

It is expected that all tasks included within **Assignment 1** will be solved **individually without any cooperation** with the other students. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or office hours. Plagiarism will result in a **FAIL** grade being recorded for the assessment task.

Please read very carefully information included in Prologue section below about software environment to be used in the subject.

Prologue

In this subject we shall use Oracle 19c database server running under Oracle Linux 7.4 operating system on a virtual machine hosted by VirtualBox. To start Oracle database server you have to start VirtualBox first. If you have not installed VirtualBox on your system yet then it is explained in Cookbook for CSIT115 Recipe 1.1, Step 1 "How to use VirtualBox ?" (`https://www.uow.edu.au/~jrg/115/cookbook/e1-1-frame.html`) how to install and how to start VirtualBox.

When VirtualBox is started, import an appliance included in a file `OracleLinux7.4-64bits-Oracle19c-22-JAN-2020.ova`. You can download `ova` image of the appliance using the links published on Moodle.

When ready, power on a virtual machine `OracleLinux7.4-64bits-Oracle19c-22-JAN-2020`.

A password to a Linux user `ORACLE` is `oracle` and a password to Oracle users `SYSTEM` and `SYS` (database administrators) is also `oracle`. Generally, whenever you are asked about a password then it is always `oracle`, unless you change it.

When logged as a Linux user, you can access Oracle database server either through a command line interface (CLI) `SQLcl` or through Graphical User Interface (GUI) `SQL Developer`.

You can find in Cookbook for CSCI317, Recipe 1, How to access Oracle 19c database server, how to use SQL Developer, how to use basic SQL and SQLcl, and how to create a sample database ?
(`https://documents.uow.edu.au/~jrg/317sim/cookbook/e1-2-frame.html`) more information on how to use `SQLcl` and `SQL Developer`.

Tasks

Task 1 (3 marks)

The objectives of this task are to learn how to create a database user, how to create a tablespace, how to create relational table in a given tablespace, how to use loader to load data into a relational table and how to find information persistent storage structures of a relational database. An additional hidden objective is to refresh your SQL skills and the ways how to use software installed on a virtual machine.

To install a sample database, perform 14 steps listed below. Note, that no report is expected from the installation of a sample database. However, please note that you will not be able to implement the assignments without a sample database.

(1) After starting a virtual machine and logging as Linux `ORACLE` user start Terminal command line shell program.

(2) Use `cd` command to move to a folder `TPCHR`.

(3) A script `tbscreate.sql` can be used to create a tablespace for the relational tables included in TPC-HR benchmark database. A conceptual schema of the database is available in a file `tpchr.pdf`.

You can use `gedit` editor (you can also use `emacs` editor) to view the contents of SQL script `tbscreate.sql`. If you like, you can also change a name of the tablespace to any valid name (for example `csci317`) and location of a file that implements the tablespace, however, it is not necessary.

(4) To connect `SQLcl` client as a user `SYSTEM` process the following command at a shell prompt.

```
sql system
```

You can also use SQL Developer client.

(5) Next, at `SQL>` prompt process a command

```
cd ~/TPCHR
```

to move a client to `TPCHR` folder.

(6) Next, process a script `tbscreate.sql` to create a new tablespace.

(7) While connected as `SYSTEM` user process the scripts `sfs.sql` and `sf.sql` to list free space per tablespace and to list the names of files implementing the tablespaces.

(8) A script `usrcreate.sql` can be used to create a new database user `tpchr` and to grant appropriate privileges and resources to the new user. Use `gedit` editor (you can

also use `emacs` editor) to view and, if you like, to update the contents of SQL script `usrcreate.sql`. To do so you have to use a command

```
!gedit usrcreate.sql
```

in front of `SQL>` prompt.

(9) Return to `SQLcl` client and process a script `usrcreate.sql`.

(10) Connect to Oracle server as a user `SYS` and process the scripts `setprivs.sql` and `setplustrace.sql` to grant appropriate privileges to a user `tpchr`.

(11) Connect to Oracle server as a new user `tpchr` and process a script `dbcreate.sql`. The script creates the relational tables of TPC-HR benchmark database.

(12) A shell script `dbload.sh` loads data into a sample database. Use an editor to view the contents of shell script `dbload.sh`. If you changed Oracle user name and/or password in step (6) then you have to update the script.

(13) Disconnect from `SQLcl` and process a shell script `dbload.sh` at a command line prompt in the following way.

```
./dbload.sh
```

and be patient, … it will take some time to load data …

(14) Connect to Oracle server as a user `tpchr` and process SQL script `dbcount.sql` to find the total number of rows in each table.

There is no need to create any reports from the steps listed above.

Implement SQL script `solution1.sql` that performs the following actions.

(1) First the script connects to a database server as a user `SYSTEM` and list the following columns from a dynamic performance view `V$INSTANCE`:
`INSTANCE_NAME`,
`HOST_NAME`,
`STARTUP_TIME`,
`DATABASE_STATUS`.

To connect to a database server within SQL script as a user `xyz007` with a password `password` insert the following line into the script.

```
connect xyz007/password
```

(2) Next, the script connects as a user `tpchr` and it processes `ANALYZE TABLE` and `ANALYZE INDEX` statements to load into a data dictionary statistical information related to the relational tables and indexes implementing a sample database `tpchr` created earlier.

(3) Next, the script retrieves and lists the following information from a data dictionary.

(i) The current timestamp obtained from an application of a function `systimestamp`.

(ii) The names of relational tables, that belong to `tpchr` sample database together with the total number of rows, total number of data blocks, total number of extents and the total number of bytes occupied by each table. Display your results in the following format.

```
table-name total-rows total-blocks total-extents total-bytes
```

(iii)The names of indexes on primary keys automatically created by the system when processing `CREATE INDEX` statements together with total number of data blocks, total number of extents and the total number of bytes occupied by each index. Display your results in the following format.

```
index-name total-blocks total-extents total-bytes
```

When ready start `SQLcl` client, connect to Oracle database server, and process SQL script `solution1.sql`. Save a report from processing of the script in a file `solution1.lst`. It is explained in Cookbook, Recipe 1.5, Step 9, "How to create and to save a report" how to save a report from processing of SQL script in a text file.

The script must be processed with `SQLcl` options `ECHO` and `FEEDBACK` set to `ON` such that all SQL statements processed are included in the report !

A good habit is to put `SQLcl` statements

```
SPOOL solution1
SET ECHO ON
SET FEEDBACK ON
SET LINESIZE 300
SET PAGESIZE 300
```

at the beginning of each SQL script implemented and the following statement at the end of the script

```
SPOOL OFF
```

A report from processing of the script must have NO syntax errors !

Deliverables
A file `solution1.lst` that contains a report from the processing of a script `solution1.sql`.

Task 2 (3 marks)
An objective of this task is to write implementation of the queries as the relational algebra expressions

Consider the following queries related to the relational tables included in `tpchr` sample database.

(1) Find the names of customers (`C_NAME`) who submitted at least one order in `2020` (`year(O_ORDERDATE)`).

(2) Find the keys of orders (`L_ORDERKEY`), that included both at least one `bolt` and at least one `screw` (`P_NAME`).

(3) Find the names of customers (`C_NAME`) who submitted no orders yet.

Write the implementations of the queries listed above as expressions of the relational algebra.

Save the relational algebra expressions implanting the queries listed above in a file `solution2.pdf`.

Deliverables
A file `solution2.pdf` that contains implementation of the queries listed above as the expressions of the relational algebra. The handwritten and scanned/photographed implementations of the queries are acceptable.

Task 3 (3 marks)
An objective of this task is to interpret a query processing plan created by a query optimizer and to draw a syntax tree of a query processing plan

Consider the following fragment of query processing plan.

```
-----------------------------------------------------------------------------------------------
| Id  | Operation               | Name          | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |               |   284K|    89M|       | 12926   (1)| 00:00:01 |
|*  1 |  HASH JOIN              |               |   284K|    89M|  7200K| 12926   (1)| 00:00:01 |
|   2 |   TABLE ACCESS FULL     | CUSTOMER      | 41861 |  6704K|       |   390   (1)| 00:00:01 |
|*  3 |   HASH JOIN ANTI        |               |   284K|    44M|    41M|  9808   (1)| 00:00:01 |
|*  4 |    TABLE ACCESS FULL    | ORDERS        |   285K|    37M|       |  2698   (1)| 00:00:01 |
|*  5 |    INDEX FAST FULL SCAN| LINEITEM_PKEY |  1943K|    48M|       |  1571   (1)| 00:00:01 |
-----------------------------------------------------------------------------------------------

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

   1 - access("O_CUSTKEY"="C_CUSTKEY")
   3 - access("ORDERS"."O_ORDERKEY"="L_ORDERKEY")
   4 - filter("ORDERS"."O_ORDERDATE"<TO_DATE(' 1996-01-01 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss') AND "O_CUSTKEY">=0)
   5 - filter("L_LINENUMBER"<10)
```

Find and draw a syntax tree of the query processing plan listed above. To draw a syntax tree, use the relational algebra operations explained during the lecture classes. Assume that the operations `HASH JOIN` and `HASH JOIN ANTI` used in a query processing plan is the same as the operations of join and antijoin in the relational algebra. Please remember, that you must create a syntax tree with the relational algebra operations explained to you during the lecture classes and NOT with the implementations of such operations by Oracle database system. Save a drawing of a syntax tree in a file `solution3.pdf`.

Deliverables
A file `solution3.pdf` with a drawing of syntax tree of the given query processing plan. A syntax tree must use the relational algebra operations explained to you during the lecture classes. You are allowed to use any line drawing tool to draw a syntax tree. A scanned/photographed copy of a neat hand drawing is also acceptable.

Task 4 (3 marks)

An objective of this task is to discover `SELECT` statement from a given query processing plan.

Consider the following fragment of query processing plan.

```
-------------------------------------------------------------------------------------
| Id  | Operation              | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |          |  1943K|   674M|       | 36788   (1)| 00:00:02 |
|*  1 |  HASH JOIN RIGHT ANTI|          |  1943K|   674M|       | 36788   (1)| 00:00:02 |
|*  2 |   TABLE ACCESS FULL  | PART     |    12 |   300 |       |   401   (1)| 00:00:01 |
|*  3 |   HASH JOIN            |          |  1943K|   628M|    41M| 36382   (1)| 00:00:02 |
|*  4 |    TABLE ACCESS FULL | ORDERS   |   285K|    37M|       |  2698   (1)| 00:00:01 |
|   5 |    TABLE ACCESS FULL | LINEITEM |  1943K|   370M|       | 12160   (1)| 00:00:01 |
-------------------------------------------------------------------------------------

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

   1 - access("LINEITEM"."L_PARTKEY"="P_PARTKEY")
   2 - filter("P_BRAND"='Golden bolts')
   3 - access("O_ORDERKEY"="L_ORDERKEY")
   4 - filter("ORDERS"."O_ORDERDATE"<TO_DATE(' 1996-01-01 00:00:00',
              'syyyy-mm-dd hh24:mi:ss'))
```

Discover `SELECT` statement that may have a query processing plan listed above. If you are not be able to get exactly the same query processing plan then, try to find `SELECT` statement with a query processing plan the most similar to the given one.

Use `EXPLAIN PLAN` statement of SQL and SQL script `showplan.sql` to find a query processing plan of a given `SELECT` statement. Save `EXPLAIN PLAN` statement and invocation of a script `showplan.sql` in SQL script `solution4.sql`.

When ready start `SQLcl` client, connect to Oracle database server, and process SQL script `solution4.sql`. Save a report from processing of the script in a file `solution4.lst`. It is explained in Cookbook, Recipe 1.5, Step 9, "How to create and to save a report" how to save a report from processing of SQL script in a text file.

The script must be processed with `SQLcl` options `ECHO` and `FEEDBACK` set to `ON` such that all SQL statements processed are included in the report !

A good habit is to put `SQLcl` statements

```
SPOOL solution4
SET ECHO ON
SET FEEDBACK ON
SET LINESIZE 300
SET PAGESIZE 300
```

at the beginning of each SQL script implemented and the following statement at the end of the script

```
SPOOL OFF
```

A report from processing of the script must have NO syntax errors !

Deliverables
A file `solution4.lst` with the outcomes from processing of `EXPLAIN  PLAN` statement and a listing of query processing plan.

Task 5 (3 marks)
An objective of this task is to find the best distribution of relational tables over the persistent storage devices.

Assume, that to avoid the conflicts with the accesses to the relational tables of TPC-HR sample database we would like to distribute the relational tables over two different persistent storage devices. Then the relational tables that are joined together can be simultaneously read from two or more persistent storage devices. Do not worry if your system does not have persistent storage devices. We shall simulate the drives through two different tablespaces `DRIVE_C` and `DRIVE_D`. You do not have to create the tablespaces. To find out, which relational tables should be located on each device we shall consider the following queries.

(i) *Find the extended prices of items (column `L_EXTENDEDPRICE`) that are available in at least a given quantity (column `PS_AVAILQTY`).*
(ii) *Find the total prices (column `O_TOTALPRICE`) of the orders that have been submitted by a customer that belongs to nation with a given name (column `N_NAME`).*
(iii) *Find the names of parts (column `P_NAME`) available in at least a given quantity (column `PS_AVAILQTY`).*
(iv) *Find the names of suppliers (column `S_NAME`) who live in a region with given region name (column `R_NAME`).*

Note, that the prefixes of the column names indicate the relational tables the columns are located at. For example, `R_NAME` denotes a column in a relational table `REGION`.

Analyze the queries listed above and find which relational tables are used by each query and distribute the relational tables over the hard drives simulated by the tablespaces `DRIVE_C` and `DRIVE_D` such, that the relational tables used by the same query are located on the different hard drives. Such approach reduces the total number of conflicts when accessing the persistent storage devices and it speeds up the query processing. If it is impossible to distribute the relational tables used by the same application on the different hard drives then try to minimize the total number of conflicts. You do not need to worry about distribution of indexes used for processing of the queries.

Create a document `solution5.pdf` that contains the following information.

(1) For each one of the queries listed above find what relational table are used by a query and draw an undirected hypergraph such that each one of its hyperedges contains the names of tables used by one query. The names of tables are the nodes of the hypergraph.

(2) Use the hypergraph created in the previous step to find distribution of the relational tables over the persistent storage devices `DRIVE_C` and `DRIVE_D` such, that the relational tables used by the same query are located on the different persistent storage devices. If it is impossible to do it locate smaller relational tables on the same device

and larger relational tables on different devices. Include information which relational table and which index is assigned to which device in a document `solution5.pdf`.

Hint
You can find a definition and visualization of an undirected hypergraph at:
`https://en.wikipedia.org/wiki/Hypergraph`

Deliverables
A file `solution5.pdf` that contains a hypergraph created in step (1) and information about relational tables assigned to the persistent storage devices. You are allowed to use any line drawing tool to draw a hypergraph. A scanned/photographed copy of a neat hand drawing is also acceptable.

Submission

Note, that you have only one submission. So, make it absolutely sure that you submit the correct files with the correct contents. No other submission is possible!

Submit the files **`solution1.lst`**, **`solution2.pdf`**, **`solution3.pdf`**, **`solution4.lst`**, and **`solution5.pdf`** through Moodle in the following way:

(1) Access Moodle at **`http://moodle.uowplatform.edu.au/`**
(2) 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
(3) When logged select a site **`CSCI317 (SP321) Database Performance Tuning`**
(4) Scroll down to a section **`Submissions`**
(5) Click at a link **`In this place you can submit the outcomes of Assignment 1`**
(6) Click at a button **`Add Submission`**
(7) Move a file **`solution1.lst`** into an area **`You can drag and drop files here to add them`**. You can also use a link **`Add`**...
(8) Repeat step (7) for the files **`solution2.pdf`**, **`solution3.pdf`**, **`solution4.lst`**, and **`solution5.pdf`**.
(9) Click at a button **`Save changes`**
(10) Click at a button **`Submit assignment`**
(11) Click at the checkbox with a text attached: **`By checking this box, I confirm that this submission is my own work,`** ... in order to confirm the authorship of your submission.
(12) Click at a button **`Continue`**

End of specification