1School of Information Technology & Computer Science
University of Wollongong

Session: **3, July 2022**
Lecturer: **Janusz R. Getta**

CSCI317 Database Performance Tuning
Singapore 2022-3
Assignment 3

Published on 6 August 2022

Scope

This assignment includes the tasks related to estimation of efficiency of indexing, horizontal partitioning of relational table, finding optimal clustering of relational tables and finding a minimal set of materialized views.

This assignment is due by **Saturday, 20 August 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 4 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 3** 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 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 (5 marks)

An objective of this task is to estimate the efficiency of indexing.

A relational table `REPAIR` contains information about the repairs performed at a network of the vehicle repair facilities.

```
REPAIR(at-address, registration, manufacturer, model,
       vehicle-type, repair-date, time-spent, description)
```

A relational table `REPAIR` has a composite primary key:
`(registration, at-address, repair-date)`.

Assume that:

(i) a relational table `REPAIR` occupies `1000` data blocks,
(ii) a relational table `REPAIR` contains `5000` rows,
(iii) an attribute `registration` has `4000` distinct values,
(iv) an attribute `manufacturer` has `50` distinct values,
(v) an attribute `vehicle-type` has `3` distinct values,
(vi) a primary key is automatically indexed,
(vii) an attribute `manufacturer` is indexed,
(viii) all indexes are implemented as B*-trees with a fanout equal to `20`,
(ix) a leaf level of an index on attribute `manufacturer` consists of `5` data blocks,
(x) a leaf level of an index on primary key consists of `100` data blocks.

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

(1)
```
SELECT DISTINCT manufacturer
FROM REPAIR;
```

(2)
```
SELECT registration
FROM REPAIR
WHERE manufacturer = 'Toyota' AND model = 'Corolla';
```

(3)
```
SELECT registration
FROM REPAIR
WHERE TO_CHAR(repair-date,'YYYY') = '2012';
```

(4)
```
SELECT COUNT(*)
FROM REPAIR
WHERE manufacturer = 'Honda';
```

(5)
```
SELECT *
FROM REPAIR
WHERE registration = 'PKR856' AND
      at-address ='15 Station St. Cooma' AND
      repair-date = '15-DEC-2010';
```

(6)
```
SELECT at-address, repair-date
FROM REPAIR
WHERE TO_CHAR(repair-date,'YYYY') = '2013;
```

(7)
```
SELECT MAX(registration)
FROM REPAIR;
```

(8)
```
SELECT TO_CHAR(repair-date,'YYYY'), COUNT(*)
FROM REPAIR
GROUP BY TO_CHAR(repair-date,'YYYY');
```

(9)
```
SELECT manufacturer
FROM REPAIR
WHERE time-spent > 20;
```

(10)
```
SELECT *
FROM REPAIR
ORDER BY manufacturer;
```

Deliverables

A file `solution1.pdf` with the comprehensive descriptions of query processing plans for each query and the estimations of the total number of read block operations needed to process each query.

Task 2 (3 marks)
An objective of this task is to implement a horizontal partitioning of relational table.

In this task we create a partitioned table `PLINEITEM` and copy the contents of a relational table `LINEITEM` into `PLINEITEM`.

A table `PLINEITEM` should be partitioned in a way that improves the performance of processing of the following classes of queries.

Find all information about the items included in the orders and such that shipment of each item happened on a given day.

Implement SQL script `solution2.sql` that performs the actions listed below.

(1) First, the script uses `EXPLAIN PLAN` statement to find a query processing plan and costs estimation for a query that belongs to a class of queries given above and such that it accesses a relational table `LINEITEM`. The script lists a query processing plan obtained from `EXPLAIN PLAN` statement.

(2) Next, the script creates a partitioned table `PLINEITEM` and loads into the table `PLINEITEM` all data from a table `LINEITEM`.

A table `PLINEITEM` should be partitioned in a way that improves the performance of processing of a class of queries listed above.

(3) Next, the script uses `EXPLAIN PLAN` statement to find a query processing plan and costs estimation for a query that belongs to a class of queries given above and such that it accesses a relational table `PLINEITEM`. The script lists a query processing plan obtained from `EXPLAIN PLAN` statement.

(4) Next, the script creates a globally partitioned index to speed up processing of the queries consistent with a given pattern.

(5) Next, the script uses `EXPLAIN PLAN` statement to find a query processing plan and costs estimation for a query that belongs to a class of queries given above and such that it accesses a relational table `PLINEITEM`. The script lists a query processing plan obtained from `EXPLAIN PLAN` statement.

(6) Next, the script drops a globally partitioned index and creates a locally partitioned index to speed up processing of the queries consistent with a given pattern.

(7) Next, the script uses `EXPLAIN PLAN` statement to find a query processing plan and costs estimation for a query that belongs to a class of queries given above and such that it accesses a relational table `PLINEITEM`. The script lists a query processing plan obtained from `EXPLAIN PLAN` statement.

(8) Finally, the script drops a locally partitioned index and partitioned relational table `PLINEITEM`.

When ready start `SQLcl` client, connect to Oracle database server, and process SQL script `solution2.sql`. Save a report from processing of the script in a file `solution2.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 solution2
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 `solution2.lst` that contains a report from the processing of a script `solution2.sql`.

Task 3 (4 marks)
Clustering
Consider a relational database created by the execution of a script `dbcreate.sql`. The database contains information about applicants for the positions advertised by employers, skills, skills possessed by applicants, skills needed for positions and skills required by other skills.

After loading data into the database the relational tables have the following sizes:

`SKILL`	10 data blocks
`SREQUIRED`	100 data blocks
`APPLICANT`	1000 data blocks
`EMPLOYER`	200 data blocks
`EMPLBY`	2000 data blocks
`POSITION`	400 data blocks
`SPOSSESSED`	500 data blocks
`SNEEDED`	300 data blocks
`APPLIES`	600 data blocks

We would like to use clustering to improve performance of the following types of queries:
(i) Find full information about the applicants who applied for a position offered by a given employer.
(ii) Find full information about the applicants who posses a give skill.
(iii) Find full information about the skills possessed by a given applicant.
(iv) Find full information about the positions applied by a given applicant.
(v) Find full information about employers who advertise more than a given number positions.

Express the queries above as `SELECT` statements.

Assume, that queries *(i)* and *(ii)* are processed 10 times per day. Assume that queries *(iii)* and *(iv)* are processed 20 times per day. Assume that query *(v)* is processed 5 times per day.

Assume that if the relational tables `r` and `s` consist of b_r and b_s blocks then their sequential scan requires b_r and b_s read block operations and their join, i.e. `r JOIN s` requires $3 * (b_r + b_s)$ read block operations.

Use a method of finding suboptimal clustering explained to you during the lecture classes in a presentation 36 Clustering relational tables to find suboptimal clustering of the sample database that improves the performance of the queries listed above.

Deliverables
A file `solution3.pdf` with the following components:
(1) `SELECT` statements implementing the queries *(i), (ii)*, *(iii)*, *(iv)* and *(v)*.
(2) Computations of costs and benefits that lead to construction of clustering graph.

(3) A drawing of a clustering graph.
(4) Suboptimal clustering that improved performance of the queries *(i), (ii)*, *(iii)*, *(iv)* and *(v)*.

Use a method of finding suboptimal clustering explained to you during the lecture classes in a presentation 17 Clustering to find suboptimal clustering of the sample database that improves the performance of the queries listed above.

Task 4 (3 marks)
Implementation of materialized views

In this task you must operate on the original state of a sample benchmark `TPC-HR` database. It is explained at the end of **Prologue** section how to return to the original state of the database.

Consider the following `SELECT` statements (also see a file `task2.sql`).

```
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;
```

An objective of this task is to create the smallest number of materialized views that can be automatically used to speed up the processing of `SELECT` statements given above.

Implement SQL script `solution4.sql` that performs the following actions.

(1) First, the script finds the query processing plans for each one of `SELECT` statements listed above. Use SQL script `showplan.sql` to list the processing plans.

(2) Next, the script creates the smallest number of materialized views that improve processing of `SELECT` statements listed above in the best way.

(3) Finally, the script again finds query processing plans the original `SELECT` statements. Use SQL script `showplan.sql` to list the processing plans.

    Of course, it is expected that the new query processing plans use the materialized view created in the previous step. It is explained in `How to … ? Cookbook, Recipe 14.2` how to set up the system initialization parameters and privileges to trigger an automatic rewriting of `SELECT` statements to a form that uses a materialized view.

(4) Finally, the script drops the materialized views created in a step (2).

When processing an improved SQL script `solution4.sql` you must put the following `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 !

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

Deliverables

A file `solution4.lst` that contains a report from the processing of a script `solution4.sql`.

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.pdf**, **solution2.lst**, **solution3.pdf**, and **solution4.lst** 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 (SP322) 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 2**
(6) Click at a button **Add Submission**
(7) Move a file **solution1.pdf** 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.lst**, **solution3.pdf**, and **solution4.lst**.
(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