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 3

Published on 31 July 2021

Scope

This assignment includes the tasks related to estimation of efficiency of indexing, finding the most efficient indexing, finding optimal clustering of relational tables, finding a minimal set of materialized views.

This assignment is due by **Saturday, 14 August 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 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.

Consider a relational table `EMPLOYEE(e#, name, salary, position)` where an attribute `e#` is a primary key.

Assume that:

(i) a relational table `EMPLOYEE` occupies 10^2 data blocks,
(ii) a relational table `EMPLOYEE` contains 10^3 rows,
(iii) an attribute name has `800` distinct values,
(iv) an attribute salary has `20` distinct values,
(v) an attribute position has `50` distinct values,
(vi) a primary key is automatically indexed,
(vii) the attributes `salary` and `position` are indexed,
(viii) all indexes are implemented as B*-trees with a fanout equal to `10`,
(ix) a leaf level of an index on attribute `salary` consists of `5` data blocks,
(x) a leaf level of an index on attribute `position` consists of `20` data blocks.

Find each of the following queries describe how a database system plans to compute the queries, (i.e. provide detailed query execution plans) and determine the total number of read block operations needed to compute the query. Show ALL computations performed to get the final answer.

(1)
```
SELECT DISTINCT salary
FROM EMPLOYEE;
```

(2)
```
SELECT position, COUNT(*)
FROM EMPLOYEE
GROUP BY position;
```

(3)
```
SELECT *
FROM EMPLOYEE
WHERE e# = 007 AND position = 'boss';
```

(4)
```
SELECT *
FROM EMPLOYEE
WHERE position = 'boss';
```

(5)
```
SELECT *
FROM EMPLOYEE
WHERE position = 'boss ' AND salary = 1000;
```

(6)
```
SELECT MAX(SALARY)
FROM EMPLOYEE;
```

(7)
```
SELECT *
FROM EMPLOYEE
WHERE position = 'boss ' OR salary = 1000;
```

(8)
```
SELECT salary
FROM EMPLOYEE;
```

(9)
```
SELECT salary, position
FROM EMPLOYEE;
```

(10)
```
SELECT *
FROM EMPLOYEE
ORDER BY salary DESC;
```

Deliverables
A file `solution4.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 improve performance of query processing through indexing.

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 relational tables included in a sample benchmark `TPC-HR` database and owned by a user `tpchr`. A conceptual schema of the database is included in a file `tpchr.png`.

We would like to use indexing to improve performance of `SELECT` statements listed below (see a file `task2.sql`). Assume, the we are not restricted by the amounts of persistent storage used for indexing and the relational tables used by the queries are not frequently modified by the data manipulation operations. It means, that for each one of the queries listed below we are allowed to create any number and any size of indexes. An objective is to improve performance of query processing through indexing as much as it is possible.

(1)
```
SELECT O_CUSTKEY, SUM(L_EXTENDEDPRICE)
FROM LINEITEM JOIN ORDERS
              ON L_ORDERKEY = O_ORDERKEY
WHERE O_ORDERSTATUS = <value-1> AND
      L_TAX = <value-2>
GROUP BY O_CUSTKEY;
```

(2)
```
SELECT C_NAME, C_PHONE
FROM CUSTOMER
WHERE C_ACCTBAL > <value-3> AND
      C_CUSTKEY NOT IN ( SELECT O_CUSTKEY
                         FROM ORDERS
                         WHERE O_TOTALPRICE > <value-4> AND
                               O_CLERK = <value-5>  );
```
(3)
```
SELECT O_ORDERKEY, O_ORDERDATE, O_TOTALPRICE
FROM ORDERS
WHERE NOT EXISTS ( SELECT 'whatever'
                   FROM LINEITEM
                   WHERE L_ORDERKEY = O_ORDERKEY AND
                         L_PARTKEY IN
                             ( SELECT P_PARTKEY
                               FROM PART
                               WHERE P_NAME = <value-6> ) );
```

The values of placeholders <value-1>, <value-2>, <value-3> , <value-4>, <value-5>, and <value-6> are up to you.

Implement SQL script solution5.sql that performs the following actions.

(1) First, the script creates the indexes to speed up query processing of SELECT statement (1).

(2) Next, the script finds and lists a query processing plan for SELECT statement (1).

(3) Next, the script drops the indexes created in a step (1).

(4) Next, the script creates the indexes to speed up query processing of SELECT statement (2).

(5) Next, the script finds and lists a query processing plan for SELECT statement (2).

(6) Next, the script drops the indexes created in a step (4).

(7) Next, the script creates the indexes to speed up query processing of SELECT statement (3).

(8) Next, the script finds and lists a query processing plan for SELECT statement (3).

(9) Next, the script drops the indexes created in a step (7).

When ready process a script solution2.sql and save a report from processing in a file solution2.lst.

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 !

You must put the following SQLcl statements

```
SPOOL solution5
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 the relational tables created by the execution of the following `CREATE TABLE` statements.

```
CREATE TABLE EMPLOYEE(
ENUM      DECIMAL(12)   NOT NULL,
FNAME     VARCHAR(50)   NOT NULL,
INITIALS  VARCHAR(5)        NULL,
LNAME     VARCHAR(50)   NOT NULL,
DOB       DATE          NULL,
BLDG      DECIMAL(3)    NOT NULL,
STREET    VARCHAR(50)   NOT NULL,
SUBURB    VARCHAR(50)   NOT NULL,
STATE     VARCHAR(5)    NOT NULL,
ZIPCODE   DECIMAL(4)    NOT NULL,
 CONSTRAINT EMPLOYEE_PKEY PRIMARY KEY(ENUM) );

CREATE TABLE DRIVER(
ENUM      DECIMAL(12)   NOT NULL,
LNUM      DECIMAL(8)    NOT NULL,
STATUS    VARCHAR(10)   NOT NULL,
 CONSTRAINT DRIVER_PKEY PRIMARY KEY(ENUM),
 CONSTRAINT DRIVER_UNIQUE UNIQUE(LNUM),
 CONSTRAINT DRIVER_FKEY FOREIGN KEY(ENUM) REFERENCES EMPLOYEE(ENUM),
 CONSTRAINT  DRIVER_STATUS  CHECK  (  STATUS  IN  ('AVAILABLE', 'BUSY', 'ON
LEAVE')) );

CREATE TABLE ADMIN(
ENUM      DECIMAL(12)    NOT NULL,
POSITION  VARCHAR(50)    NOT NULL,
 CONSTRAINT ADMIN_PKEY PRIMARY KEY(ENUM),
 CONSTRAINT ADMIN_FKEY FOREIGN KEY(ENUM) REFERENCES EMPLOYEE(ENUM) );

CREATE TABLE TRUCK(
REGNUM    VARCHAR(10)   NOT NULL,
CAPACITY  DECIMAL(7)    NOT NULL,
WEIGHT    DECIMAL(5)    NOT NULL,
STATUS    VARCHAR(10)   NOT NULL,
 CONSTRAINT TRUCK_PKEY PRIMARY KEY(REGNUM),
 CONSTRAINT TRUCK_STATUS CHECK (STATUS IN ('AVAILABLE', 'USED',
                                                    'MAINTAINED')) );
CREATE TABLE TRIP(
TNUM      DECIMAL(10)   NOT NULL,
LNUM      DECIMAL(8)    NOT NULL,
REGNUM    VARCHAR(10)   NOT NULL,
TRIP_DATE DATE          NOT NULL,
 CONSTRAINT TRIP_PKEY PRIMARY KEY (TNUM),
 CONSTRAINT TRIP_FKEY1 FOREIGN KEY (LNUM) REFERENCES DRIVER(LNUM),
 CONSTRAINT TRIP_FKEY2 FOREIGN KEY (REGNUM) REFERENCES TRUCK(REGNUM) );

CREATE TABLE TRIPLEG(
TNUM      DECIMAL(10)   NOT NULL,
LEGNUM    DECIMAL(2)    NOT NULL,
```

```
DEPARTURE VARCHAR(30)   NOT NULL,
DESTINATION VARCHAR(30) NOT NULL,
CONSTRAINT TRIPLEG_PKEY PRIMARY KEY (TNUM, LEGNUM),
 CONSTRAINT TRIPLEG_UNIQUE UNIQUE(TNUM, DEPARTURE, DESTINATION),
 CONSTRAINT TRIPLEG_FKEY1 FOREIGN KEY (TNUM) REFERENCES TRIP(TNUM) );
```

The database contains information about employees, drivers and administration staff, trucks, trips made by drivers, and legs of each trip.

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

EMPLOYEE	60 data blocks
DRIVER	30 data blocks
ADMIN	10 data blocks
TRUCK	50 data blocks
TRIP	100 data blocks
TRIPLEG	300 data blocks

We would like to use clustering to improve performance of the following types of queries:

(i) Find full information about the drivers who live at a given address.
(ii) Find full information about the administration people who live at a given address.
(iii) Find full information about the trucks used by a driver with a given license number.
(iv) Find full information about the drivers who made a trip on a given date.
(v) Find full information about the legs of trips that used a truck with a given registration number.

Assume, that queries *(i)* and *(ii)* are processed 10 times per day. Assume that queries *(iii)* and *(iv)* are processed 30 times per day. Assume that query *(v)* is processed 20 times per day.

Assume that the relational tables `r` and `s` consist of b_r and b_s blocks each. Then

- if `r` and `s` are clustered together then to read a cluster we need $b_r + b_s$ read block operations and
- if `r` and `s` are not clustered together then to join the tables we need $3*(b_r + b_s)$ read block operations (approximate estimation of hash-based join).

Use a method of finding suboptimal clustering explained to you during the lecture classes in a presentation 18 Clustering 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) Computations of costs and benefits that lead to construction of clustering graph.
(2) A drawing of a clustering graph.
(3) Optimal clustering that improves performance of some of some of queries *(i), (ii)*, *(iii)*, *(iv)* and *(v)*.

Task 4 (3 marks)
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 (see a file `task4.sql`).

```
SELECT L_PARTKEY, L_SUPPKEY, SUM(L_QUANTITY)
FROM LINEITEM
GROUP BY L_PARTKEY, L_SUPPKEY;

SELECT L_SUPPKEY, L_ORDERKEY, SUM(L_DISCOUNT)
FROM LINEITEM
GROUP BY L_SUPPKEY, L_ORDERKEY;

SELECT L_SUPPKEY, L_ORDERKEY, SUM(L_QUANTITY)
FROM LINEITEM
GROUP BY L_SUPPKEY, L_ORDERKEY;

SELECT L_PARTKEY, L_SUPPKEY, COUNT(L_DISCOUNT)
FROM LINEITEM
GROUP BY L_PARTKEY, L_SUPPKEY;

SELECT L_PARTKEY, L_SUPPKEY, SUM(L_DISCOUNT)
FROM LINEITEM
GROUP BY L_PARTKEY, L_SUPPKEY;

SELECT L_SUPPKEY, COUNT(DISTINCT L_DISCOUNT)
FROM LINEITEM
GROUP BY L_SUPPKEY;

SELECT L_PARTKEY, L_SUPPKEY, COUNT(DISTINCT L_DISCOUNT)
FROM LINEITEM
GROUP BY L_PARTKEY, L_SUPPKEY;
```

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 lists the query processing plans for each one of `SELECT` statements listed above.

(2) Next, the script creates the smallest number of materialized views that improve processing of `SELECT` statement listed above in the best way. It is recommended to find in Cookbook information on how to do it.

(3) Finally, the script again lists the query processing plans of the original `SELECT` statements. Of course, it is expected that the new query processing plans will use the materialized view created in the previous step.

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

When ready process a script `solution4.sql` and save a report from processing in a file `solution4.lst`.

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 !

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 !

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 (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 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