CS 460 – Database Management Systems

Winter semester 2020

3rd Assignment
Due date: 23/12/2020

General instructions

Write a report demonstrating your results using images/screenshots, SQL code, SQL errors/warnings and any other explanation necessary. Include a cover page with your student ID and name. The filename of the deliverable must be in the following form: **ask3_AM** (where AM is your student id number e.g. ask3_1234). Any images/screenshots should be included in your report document. Do NOT submit separate image files. The report document must be in pdf format. Submit a single compressed file (zip/rar/gz etc) if your work contains multiple files. Note that this assignment is to be done **individually**. Cheating in any way, including giving your work to someone else will result in failing this assignment (a mark of zero will be given).

Submit your assignment **electronically** (online) until 29/11/2019 at 23:59 using the course webpage at the eLearn system.

Overview

In this assignment, you will experiment with SQL indexing structures using various configurations. You will be working with the latest Oracle Database Express Edition.

Background

We strongly suggest you read/review the following resources:

- Course slides and tutorials on Query Execution Optimization

Additional Info. The Oracle optimizer was built to make your tuning life easier by choosing better paths for your poorly written queries. Rule-based optimization (now obsolete and unsupported) was built on a set of rules for how Oracle processes statements. Oracle 10g Release 2 only supported the use of the cost-based optimizer; the rule-based optimizer was no longer supported. Since Oracle 10g Release 2, Oracle has automatic statistics gathering turned on to aid the effectiveness of the cost-based optimizer. In Oracle, many features are only available when using cost-based optimization. Rule-based optimization is Oracle-centric, whereas cost-based optimization is data-centric. The optimizer mode under which the database operates is set via the initialization parameter OPTIMIZER_MODE. The possible optimizer modes are as follows:

- **ALL_ROWS** Gets all rows faster (generally, forces index suppression). This is good for untuned, high-volume batch systems. This is the default.
- **FIRST_ROWS** Gets the first row faster (generally forces index use). This is good for untuned systems that process lots of single transactions.
- **FIRST_ROWS (1|10|100|1000)** Gets the first n rows faster. This is good for applications that routinely display partial results to users such as paging data to a user in a web application.
- **CHOOSE** Now obsolete and unsupported but still allowed. Uses cost-based optimization for all analyzed tables. This is a good mode for well-built and well-tuned systems (for advanced users). This option is not documented for 11gR2 and beyond but is still usable.
- **RULE** Now obsolete and unsupported but still allowed. Always uses rule-based optimization. If you are still using this, you need to start using cost-based optimization, as rule-based optimization is no longer supported under Oracle 10g Release 2 and higher.
- The default optimizer mode for Oracle 11g Release 2 is ALL_ROWS. Also, cost-based optimization is used even if the tables are not analyzed.

Note! Remember to update statistics by yourself when cost based optimization is used!

Set-up

Use the same dataset as in Assignment 1. Create the following tables (tables.sql) and insert the data in the accompanying files (distributedBy.sql – 143.153 tuples, movie.sql – 193.781 tuples, people.sql - 289.221 tuples, plays.sql – 810.692 tuples, producedBy.sql – 202.868 tuples). Initially there are not defined indexes, keys or clusters for the aforementioned tables. In addition, create the table PLAN_TABLE, where you will save the execution plans of each assignment's query.

Task 1 – Cost based optimization

Use the command ALTER SESSION to set the parameter OPTIMIZER_MODE to ALL_ROWS. Using this parameter value the optimizer is set to use a cost based optimization method, trying to identify the best resource allocation for query execution.

A) Create a hash cluster for the *movieID* using 1000 search keys and relate it with the *movie* table.
To do that you should drop and re-create the movie table to take into advantage of the hash cluster.

B) Find the execution plan of the following queries. What observation do you make?

```
SELECT      movieTitle
FROM        movie
WHERE       year > 1990;
```

```
SELECT      personName
FROM        people
WHERE       birthYear > 1945;
```

C) Next, create an unclustered B^+-Tree for the *year* field of the table *movie* and one for the *birthYear* of the table *people*. Which is now the execution plan of the following SQL queries?

E) Find the execution plan for the following queries for the table movie including the fields *movieID* and *year* in the WHERE clauses. Which one of the hash cluster on the *movieID* and the B^+-Tree on the field *year* are being used? What are the reasons for the possible different execution plans?

```
SELECT      movieID, movieTitle
FROM        movie
WHERE       movieID != 0046778 and year > 1985;
```

```
SELECT      movieID, movieTitle
FROM        movie
WHERE       movieID = 0046778 and year > 1985;
```

Task 2 – Rule based optimization

Up to this point you should have available a hash cluster on the *movieID* used by the *movie* and the *plays* tables. In addition, there should be available two unclustered B^+-Tree indexes for the *year* and the *birthYear* fields of the *movie* and *people* tables respectively.

A) Use the command ALTER SESSION to set the OPTIMIZER_MODE to RULE. Using this command the optimizer uses a rule based optimization method ignoring statistics.

B) Now execute again the (B) and (E) of the Task 1. Is there any difference in the execution plan? What about the execution times?

B) Define for the *movieTitle* field a unique unclustered B^+-Tree. Find the query execution plan for the following query. Is the hash cluster used or the unclustered B^+-Tree. Why?

```
SELECT m.movieID, m.movieTitle, p.companyID
FROM movie m, producedBy p
WHERE m.movieID = p.movieID and m.movieID = 0046799 and m.movieTitle = 'Boot Polish
(1954)';
```

Now drop the index on the *movieTitle*.

Task 3 – Compare optimization methods

Up to this point you should have available a hash cluster on the field *movieID* used by the *movie* and the *plays* tables. In addition, there are two unclustered B^+-Tree indexes for the *year* and the *birthYear* fields of the *movie* and the *people* tables respectively. Finally, the optimizer_mode is set to the RULE value.

A) For each one of the OPTIMIZATION_MODE values (RULE, FIRST_ROWS, ALL_ROWS) find the execution plans and the execution times of the following query.:

```
SELECT /*+ ORDERED USE_HASH(p,pl) */ pl.movieID
FROM people p, plays pl
WHERE p.personID = pl.personID and p.birthYear > 1990;
```

B) Drop the index for the *birthYear* field and repeat (A). For the same mode do you observe any differentiation between the two queries?

C) Create a bitmap index on the *birthYear* field. Repeat (A). What differences do you observe using the bitmap index?

D) Compare the execution times for (A), (B) and (C) for each optimization mode. What differences do you observe?

E) Drop the *people* table and recreate it, defining a clustered B^+-Tree on the *birthYear* field. For each one of the three OPTIMIZATION_MODE (RULE, FIRST_ROWS, ALL_ROWS) find the execution plan and the execution time for the query in (A). Do you find any difference now when compared to the unclustered B^+-Tree index?