# Advanced Database

# Spring 2015

# Indexing

|  |  |
| --- | --- |
| **Submission Format:** | Online on D2L Dropbox as a PDF document |

The tasks for this deliverable are described below. All work must be done in your database.

**1. Copy Database**

Download the file, createTablesWOIdx.sql (Create Populated Tables File link), from D2L. In this file is the SQL necessary to create a database complete with tables populated with data. Using SQL Developer, connect to your database. From the SQL Developer menu, select File -> Open and navigate to where you downloaded createTablesWOIdx.sql and open that file. You should see the SQL code in your SQL editor. Click the Run Script button. After this has run through completion, view the log. The only errors you should see in the log are DROP TABLE failures. This can be caused by a couple of things. It is possible you didn’t have the table to start with. You also could have created the table and misspelled the table name when you created it. Do not be alarmed if you get these failure messages. You should see where the 10 tables were successfully created as well as the tables altered after adding primary keys and other constraints. At the end of this document is a table diagram. You may find it useful to explore the database before you start. Look at the data types of the columns. Look at some of the rows. You will be able to see the different domains for the columns.

**2. Verify Data**

Explore the new database and record the number of records for each table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Order** | **Table** | **Records** |  | **Order** | **Table** | **Records** |
| 1 | STUDENT | 10,369 |  | 6 | ACTIVITY | 77,726 |
| 2 | SCHOOL | 7,160 |  | 7 | SCHOOL\_APPLY | 39,488 |
| 3 | GUARDIAN | 10,514 |  | 8 | SCHOOL\_ATTEND | 9,568 |
| 4 | EXAM | 3 |  | 9 | STUDENT\_GUARDIAN | 13,600 |
| 5 | VOLUNTEER\_WORK | 43,331 |  | 10 | STUDENT\_EXAM | 23,329 |

**3. Tune the Queries**

Attachment 1, table diagram, will help you complete these tasks. The goal of tasks 1 – 6 is for you to tune the database using indexes to achieve the best performance from the DB in order to reduce the cost of running the queries to the lowest cost you can achieve. You should try ONE thing at a time. Otherwise, you won’t know the true savings of your attempt. List what you did and the total cost for running the query using the tuning mechanism you selected. For example, if you build an index on a field in the table, list that you created an index, type of index, as well as the table and column and list the total cost. Not every attempt will reduce the cost but still list what you tried even if there was not a cost savings. You will need to implement multiple performance mechanisms, not just one. Ensure you start each task with the base line cost, meaning, you may have to delete indexes created on previous steps as not to falsely affect the baseline result.

Note: The script has three queries for creating indices (IDX\_GUARDIAN\_NAME, IDX\_STUDENT\_ENROLL\_DATE and IDX\_SCHOOL\_NAME)

1. Enter the following query into SQL Developer.

SELECT COUNT(\*), state

FROM school

GROUP BY state

ORDER BY COUNT(\*)

;

Base Line Cost: 18

After creating index for state (IDX\_SCHOOL\_STATE) the new cost: 6



SELECT count(\*), city

FROM guardian

GROUP BY city

ORDER BY city

;

Base line cost: 33

After creating index for city (IDX\_GUARDIAN\_CITY) the new cost: 10

3.

SELECT last\_name, description

FROM student

JOIN activity USING (student\_id)

WHERE gender = 'F' AND

grade\_level = '8' AND

start\_date <= '03-APR-06'

;

Base line cost: 180

After creating index for last\_name (IDX\_STUDENT\_LAST\_NAME) the new cost: 180 (no change)

After creating index for grade\_level (IDX\_STUDENT\_GRADE\_LEVEL) the new cost: 180 (no change)

After creating index for student\_id (IDX\_ACTIVITY\_STUDENT\_ID) the new cost: 180 (no change)

With three separate indices i.e.; IDX\_STUDENT\_GENDER (bitmap), IDX\_STUDENT\_GRADE\_LEVEL (Non-Unique), IDX\_STUDENT\_START\_DATE the new cost: 150

4.

SELECT last\_name || ', ' || first\_name AS name, COUNT(\*) AS num\_accepted

FROM student s

JOIN school\_apply sa ON (s.student\_id = sa.student\_id)

WHERE decision = 'accepted' AND

grade\_level = 8

GROUP BY s.last\_name, s.first\_name

HAVING count(\*) <= 2

ORDER BY count(\*) DESC, last\_name, first\_name;

Base line cost: 87

After creating index for student\_id (IDX\_SCHOOL\_APPLY\_STUDENT\_ID) the new cost: 87 (no change)

After creating index for first\_name, last\_name (IDX\_STUDENT\_FIRST\_LAST\_NAME) the new cost: 87 (no change)

After creating index for school\_id (IDX\_SCHOOL\_APPLY\_SCHOOL\_ID) the new cost: 87 (no change)

5.

SELECT last\_name, first\_name, name

FROM student s JOIN school\_attend a USING (student\_id)

JOIN school c USING (school\_id)

WHERE s.end\_date >= '01-JAN-06'

AND s.end\_reason = 'Graduated'

AND c.state = 'GA'

AND type = 'C'

AND a.end\_date IS NULL

ORDER BY name, last\_name, first\_name;

Baseline Cost: 82

After creating index for state (IDX\_SCHOOL\_STATE) the new cost: 82 (no change)

6. Create a Bitmap Index on the Student table on the Gender column. Execute a query that selects all of the female students and displays their last name and first name.

Does this query utilize your bitmap index?

Based on your answer above, explain why you think it does or doesn’t.

Query:

SELECT s.last\_name, s.first\_name

FROM student s

WHERE s.gender= 'F';

Baseline Cost: 48

Cost (After creating bitmap on gender) : 48

It did not utilize the bitmap index, probably because the full table scan was probably faster than using the bitmap.

7. Ensure you have submitted your assignment to D2L. Many times, you have loaded your assignment, however, you have not submitted and I, therefore, do not have access to it. Also, ensure the document you submit to D2L can be downloaded and opened. If for whatever reason, I either cannot download your document or cannot open it, that is the same as not submitting it and you will receive a zero for the assignment.

