CSE 135 Project Indices Report

Please note: The queries utilized by the Sales Analytics Page can be referred to on *SalesDAO.java* and *ProductDAO.java*. A simplified compilation was also created in the sqlScripts folder titled *sql\_sales\_analysis\_queries.sql*. Each query is labelled by a number according to that .sql file and will be referred to as such in the following.

**Listing of Possibly Beneficial Indices**:

For the following queries, it might be beneficial to index on…

*Query 1*:

* state table’s state\_name
* person table’s state\_id
* products\_in\_cart table’s product\_id
  + cart\_id
* shopping\_cart table’s person\_id
* product table’s category\_id

*Query 2*:

* products\_in\_cart table’s product\_id

*Query 3*: This is basically Query 2 but taking all products even if they have null prices thus the same index applies. It also applies a filter thus it might be simplified with an index on:

* product table’s category\_id

*Query 4*:

* person table’s state\_id
* products\_in\_cart table’s product\_id
  + cart\_id
* shopping\_cart table’s person\_id

*Query 5*:

* shopping\_cart table’s person\_id
* products\_in\_cart table’s product\_id
  + cart\_id

*Query 6*: The above query along with:

* product table’s category\_id

*Query 7*:

* shopping\_cart table’s person\_id
* products\_in\_cart table’s product\_id
  + cart\_id

**Index Testing:** Notes: Referring to indices listed in *indices.sql*. Testing was done via pgadmin and the data generator was used for each case. For Case 1 (small and hot), I generated 20 customers, 15 categories, 15 products, and 20 sales. For Case 2 (large and cold), I generated 5000 customers, 2500 categories, 150,000 products, and 450,000 sales. I averaged the times by taking the minimum 5 out of 10 runs for case 1. For the cold case, I only averaged over the first 3 runs. Each query was run without additional filtering (ie. category\_id) except for index 6 which involves indexing category\_id. There, I used a default value of 8 for timing tests.

For the full running time, I ran each query twice (varying queries to clear the cache for the cold case) and averaged over all collected times (around 7 to 8).

*Code*: I ran corresponding *indices.sql* and *sql\_sales\_analysis\_queries.sql* on data generated as above for the following tests. For the running time of the jsp, I called System.nanoTime() for the salesAnalytics.jsp page and found the difference before and after rendering the table.

*No Indexing:*

Case 1:

a) running time = 7 ms

b) individual query times:

i) 821.4 ms

ii) 656.7 ms

iii) 648.2 ms

iv) 621 ms

v) 646.4 ms

vi) 594 ms

vii) 614 ms

Case 2:

a) running time = 67,782.81 ms

b) individual query

i) 789.5 ms

ii) 1 sec

iii) 1 sec

iv) 3 sec

v) 917 ms

vi) 878.3 ms

vii) 3.3 sec

*Index 1:*

Case 1:

a) running time = 5 ms

b) individual query times:

i) 680.6 ms

~~ii) 764.2 ms~~ Query 2 should not be influenced by state

~~iii) 676 ms~~ Should not be influenced by state

iv) 612 ms

v) 540.8 ms

vi) 678.2 ms

vii) 615 ms

Case 2:

a) running time = 195 ms

b) individual query

i)

~~ii)~~

~~iii)~~

iv)

v)

vi)

vii)

*Index 2:*

Case 1:

a) running time = 6 ms

b) individual query

i) 567.6 ms

ii) 651 ms

iii) 635.4 ms

iv) 563.2 ms

v) 598.6 ms

vi) 649.8 ms

vii) 690.8 ms

Case 2:

a) running time

b) individual query

i)

ii)

iii)

iv)

v)

vi)

vii)

*Index 3:*

Case 1:

a) running time = 4 ms

b) individual query

i) 586.2 ms

ii) 604.2 ms

iii) 551.2 ms

iv) 782.2 ms

v) 722.6 ms

vi) 683.4 ms

vii) 693.2 ms

Case 2:

a) running time

b) individual query

i)

ii)

iii)

iv)

v)

vi)

vii)

*Index 4:*

Case 1:

a) running time = 4 ms

b) individual query

i) 583 ms

ii) 665.2 ms

iii) 705.4 ms

iv) 676.4 ms

v) 653.6 ms

vi) 664.4 ms

vii) 724.8 ms

Case 2:

a) running time

b) individual query

i)

ii)

iii)

iv)

v)

vi)

vii)

*Index 5:*

Case 1:

a) running time = 4 ms

b) individual query

i) 653.4 ms

ii) 641.4 ms

iii) 607.4 ms

iv) 641.6 ms

v) 606.6 ms

vi) 603 ms

vii) 660.6 ms

Case 2:

a) running time

b) individual query

i)

ii)

iii)

iv)

v)

vi)

vii)

*Index 6:* (Using category\_id 8 as a filter)

Case 1:

a) running time = 3 ms

b) individual query

i) 605.2 ms

ii) 703.8 ms

iii) 632.4 ms

iv) 520.6 ms

v) 696.6 ms

vi) 632.4 ms

vii) 556.6 ms

Case 2:

a) running time

b) individual query

i)

ii)

iii)

iv)

v)

vi)

vii)

**Conclusion:** The best index choice(s)

o For each of the two cases report (a) running time of the overall jsp, (b) running time of the individual queries and (c) your index choices for these two cases.

▪ In the two extreme cases, adjust the Postgres sequential-Vs-random access ratio  
accordingly.  
  
o For each index Y that you are sure that it is beneficial, you need not contact any  
experiment to verify its usefulness. Just mark that you are sure and provide your  
reasoning. Nevertheless, debug: Use EXPLAIN to check whether some query (or queries)  
of your program indeed used the index Y

o Next, for each candidate index in the indices’ report where you are not sure whether it  
actually benefits performance, do an experiment. The simple way to experiment if an  
index *X* is beneficial is to first run the “Sales Analytics” page without having created the  
index *X* and then create the index *X* and run the page again. If you see no performance  
difference, the index was not worthy. Be careful of caching effects when you execute  
such experiments. It is very possible that an index may useful in the “small and hot” but  
not in the “large and cold” and vice versa.  
  
  
  
  
• You will need to provide the following and argue for the appropriateness of your solution:  
o Best index choice.  
o The indices’ report: candidate indices, which one of them are the indices you are sure  
that are beneficial (and your reasoning), which ones you experimented with in order to  
decide whether they are (or they are not) useful, the experimentation results.