Assignment 2

Shubham Garg, 2010 CE 10411
Sumeet Kumar Sinha, 2010 CE 10405
Due date: March 18, 2013, 11:55pm IST

Each query takes a different amount time to run on the default database. Explain why this is the case.

- In Query 1 we are supposed to search for condition when object = '<United_States>'
  - There are roughly 150000 tuples where object='<United_States>'
  - Whereas in Query 2 we are supposed to search for condition when object = '<Morocco>'
  - There are only about 3000 tuples where object='<Morocco>'
- Hence in Query 1 more time is taken by the processor as the condition subject <> object needs to be checked on more tuples in Query 1 than in Query 2
- In Query 3 we are supposed to search for both hence the time taken is even greater than Query 1 but not significantly greater as Query 2 takes very little time
- In Query 4 no condition is specified with respect to the object hence it takes the max. time because the remaining 2 conditions have to be checked on the entire database

Explanation for each optimisation why it performed better or worse. Prove that you performed An optimisation by pasting the query plan generated. Note that some optimizations, may not Generate a new plan. In these cases, you should show the query plan, and explain why the Optimisation did not make a deference.

Query 1:

- normal:bitmapscan on:(bitmap heap scan) (bitmap index scan objectpredicate) 1640 warm
- bitmapscan on:(bitmap heap scan) (bitmap index scan object) 1630 warm
- bitmapscan on:(bitmap heap scan) (bitmap index scan predicate) 7719 warm
- bitmapscan off:(index scan objectpredicate) 626 warm
- bitmapscan off:(index scan object) 660 warm
- bitmapscan off:(index scan predicate) 6194 warm
- bitmap off index off: (seq. scan) 4282 warm
- if all things are off it still uses index scan
- by default it uses bitmap even if geqo_effort is increased or decreased

Default Query Plan:

In the default case bitmap scan is used in which firstly the heap scan is done on yagofacts followed by index scan on index objectpredicate and 1640ms is taken.

Query Plan:
GroupAggregate (cost=4643.53...198478.29 rows=548 width=42)
-> Bitmap Heap Scan on yagofacts (cost=4643.53...197848.59 rows=124845 width=42)
Recheck Cond: (((object): text = '<United_States>': text) AND ((predicate): text = '<links to>': text))
Filter: ((subject): text <> (object): text)
-> Bitmap Index Scan on yagoindexobjectpredicate (cost=0.00..4612.32 rows=125473 width=0)

Index Cond: (((object)::text = '<United_States>'::text) AND ((predicate)::text = '<linksTo>'::text))

Whereas if we turn off bitmap scan using:

set enable_bitmapscan=false;

Then only index scan on index objectpredicate is done and following query plan is generated:

Query Plan:
GroupAggregate (cost=0.00..484820.38 rows=548 width=42)
-> Index Scan using yagoindexobjectpredicate on yagofacts (cost=0.00..484190.67 rows=124845 width=42)
Index Cond: (((object)::text = '<United_States>'::text) AND ((predicate)::text = '<linksTo>'::text))
Filter: ((subject)::text <> (object)::text)

Here the time taken is only 626ms as in bitmap scan heap scan was also being performed with no real benefit and hence it was a redundancy. In we don't have objectpredicate index and instead only have an index on object then the following query plan is generated

Query Plan:
GroupAggregate (cost=0.00..553974.82 rows=548 width=42)
-> Index Scan using yagoindexobject on yagofacts (cost=0.00..553345.12 rows=124845 width=42)
Index Cond: ((object)::text = '<United_States>'::text)
Filter: (((subject)::text <> (object)::text) AND ((predicate)::text = '<linksTo>'::text))

And the time taken is slightly higher as now we don't have an index on predicate.
However if instead of object we have index only on predicate then following plan is generated(if bitmap scan is off)

Query Plan:
GroupAggregate (cost=0.00..35680145.05 rows=548 width=42)
-> Index Scan using yagoindexpredicate on yagofacts (cost=0.00..35679515.35 rows=124845 width=42)
Index Cond: ((predicate)::text = '<linksTo>'::text)
Filter: (((subject)::text <> (object)::text) AND ((object)::text = '<United_States>'::text))

And the time taken is 6194ms approx. 10 times more as there are few tuples where object= '<United_States>' compared to case where predicate='<linksTo>' hence sequential scanning on object takes more time.

And if bitmap is On then query plan generated is

Query Plan:
GroupAggregate (cost=4043.99..207835.40 rows=548 width=42)
-> Bitmap Heap Scan on yagofacts (cost=4043.99..207205.70 rows=124845 width=42)
Recheck Cond: ((object)::text = '<United_States>'::text)
Filter: (((subject)::text <> (object)::text) AND ((predicate)::text = '<linksTo>'::text))
-> Bitmap Index Scan on yagoindexobject (cost=0.00..4012.77 rows=144598 width=0)
Index Cond: ((object)::text = '<United_States>'::text)

And the time taken is 7719 as here in addition to index scan on index predicate heap scan is also performed

If even the index on predicate is removed then sequential scanning is done with following query plan

Query Plan:
GroupAggregate (cost=0.00..562497.95 rows=548 width=42)
-> Seq Scan on yagofacts (cost=0.00..561868.24 rows=124845 width=42)

Filter: (((subject)::text <> (object)::text) AND ((object)::text = '<United_States>'::text) AND ((predicate)::text = '<linksTo>'::text))

And the time taken is 4282 which shows that in this case indexing on only predicate is useless and costs even more than seq. scan.

So for Query 1 the best optimization plan is index scan with indexing on (object, predicate) and the worst case is when we force it to use bitmap scan with indexing done only on predicate.

Query 2:

- normal:bitmapscan on:(bitmap heap scan) (bitmap index scan objectpredicate) 3.516 warm
- bitmapscan on:(bitmap heap scan) (bitmap index scan object) 4.486 warm
- bitmapscan on:(bitmap heap scan) (bitmap index scan predicate) 7698 warm
- bitmapscan off:(index scan objectpredicate) 3.508 warm
- bitmapscan off:(index scan object) 5.303 warm
- bitmapscan off:(index scan predicate) 6091 warm
- bitmap off index off: (seq. scan) 4203 warm
- if all things are off it still uses index scan
- by default it uses bitmap even if geqo_effort is increased or decreased

Default Query Plan:

In the default case bitmap scan is used in which firstly the heap scan is done on yagofacts followed by index scan on index objectpredicate and 3.516ms is taken.

Query Plan:
GroupAggregate (cost=15.46..735.46 rows=1 width=42)
-> Bitmap Heap Scan on yagofacts (cost=15.46..734.54 rows=182 width=42)
Recheck Cond: (((object)::text = '<Morocco>'::text) AND ((predicate)::text = '<linksTo>'::text))
Filter: ((subject)::text <> (object)::text)
-> Bitmap Index Scan on yagoindexobjectpredicate (cost=0.00..15.42 rows=183 width=0)
Index Cond: (((object)::text = '<Morocco>'::text) AND ((predicate)::text = '<linksTo>'::text))

Whereas if we turn off bitmap scan using:

set enable_bitmapscan=false;

Then only index scan on index objectpredicate is done and following query plan is generated:

Query Plan:
GroupAggregate (cost=0.00..749.02 rows=1 width=42)
-> Index Scan using yagoindexobjectpredicate on yagofacts (cost=0.00..748.10 rows=182 width=42)
Index Cond: (((object)::text = '<Morocco>'::text) AND ((predicate)::text = '<linksTo>'::text))
Filter: ((subject)::text <> (object)::text)

Here the time taken is only 3.508ms as in bitmap scan heap scan was also being performed with no real benefit and hence it was a redundancy. In we don't have objectpredicate index and instead only have an index on object then time taken is slightly higher as now we don't have an index on predicate.
However if instead of object we have index only on predicate then following plan is generated(if bitmap scan is off)

Query Plan:
GroupAggregate (cost=0.00..35679516.27 rows=1 width=42)
-> Index Scan using yagoindexpredicate on yagofacts (cost=0.00..35679515.35 rows=182 width=42)
Index Cond: ((predicate)::text = '<linksTo>'::text)
Filter: (((subject)::text <> (object)::text) AND ((object)::text = '<Morocco>'::text))

And the time taken is 6091ms approx. 10 times more as there are few tuples where object= '<United_States>' compared to case where predicate='<linksTo>' hence sequential scanning on object takes more time.

And if bitmap is On then query plan generated is

Query Plan:
GroupAggregate (cost=13.92..842.56 rows=1 width=42)
-> Bitmap Heap Scan on yagofacts (cost=13.92..841.64 rows=182 width=42)
Recheck Cond: ((object)::text = '<Morocco>'::text)
Filter: (((subject)::text <> (object)::text) AND ((predicate)::text = '<linksTo>'::text))
-> Bitmap Index Scan on yagoindexobject (cost=0.00..13.87 rows=211 width=0)
Index Cond: ((object)::text = '<Morocco>'::text)

And the time taken is 7698 as here in addition to index scan on index predicate heap scan is also performed

If even the index on predicate is removed then sequential scanning is done with following query plan

Query Plan:
GroupAggregate (cost=0.00..561869.16 rows=1 width=42)
-> Seq Scan on yagofacts (cost=0.00..561868.24 rows=182 width=42)
Filter: (((subject)::text <> (object)::text) AND ((object)::text = '<Morocco>'::text) AND ((predicate)::text = '<linksTo>'::text))

And the time taken is 4203 which shows that in this case indexing on only predicate is useless and costs even more than seq. scan.

So for Query 2 as was for Query 1 the best optimization plan is index scan with indexing on (object, predicate) and the worst case is when we force it to use bitmap scan with indexing done only on predicate.

Query 3:

- normal:bitmapscan on:(bitmap heap scan) (bitmap or) (bitmap index scan 2 times object and objectpredicate) 2040 warm
- bitmapscan on:(bitmap heap scan) (bitmap or) (bitmap index scan 2 times object both) 2033 warm
- bitmapscan off index on seq off:(index scan object) 138699 warm
- bitmap off index on: (seq. scan) 6200 warm
- if all things are off it still uses index scan
- if object and objectpredicate is removed then seq scan is done even if it is off

Default Query Plan:

In Query 3 by default bitmap scan is done with following query plan

Query Plan:
HashAggregate (cost=208431.15..208437.50 rows=635 width=42)
-> Bitmap Heap Scan on yagofacts (cost=4100.58..207707.25 rows=144779 width=42)

Recheck Cond: ((((object)::text = '<Morocco>'::text) AND ((predicate)::text = '<linksTo>'::text)) OR ((object)::text = '<United_States>'::text))
Filter: ((((subject)::text <> (object)::text) AND ((predicate)::text = '<linksTo>'::text) AND ((object)::text = '<Morocco>'::text)) OR ((object)::text = '<United_States>'::text))
-> BitmapOr (cost=4100.58..4100.58 rows=144781 width=0)
-> Bitmap Index Scan on yagoindexobjectpredicate (cost=0.00..15.42 rows=183 width=0)
Index Cond: (((object)::text = '<Morocco>'::text) AND ((predicate)::text = '<linksTo>'::text))
-> Bitmap Index Scan on yagoindexobject (cost=0.00..4012.77 rows=144598 width=0)
Index Cond: ((object)::text = '<United_States>'::text)

and the time taken is 2040 ms.

If bitmap scan is turned off then seq scan is performed with following query plan:

Query Plan:
GroupAggregate (cost=0.00..76386100.59 rows=635 width=42)
-> Index Scan using yagoindexobject on yagofacts (cost=0.00..76385370.34 rows=144779 width=42)
Filter: ((((subject)::text <> (object)::text) AND ((predicate)::text = '<linksTo>'::text) AND ((object)::text = '<Morocco>'::text)) OR ((object)::text = '<United_States>'::text))

And the time taken is 6200 ms which is as expected is greater than bitmap scan as scanning is done without indexing
If seq scan is also turned off then index scan is done on index object with following plan:

Query Plan:
GroupAggregate (cost=0.00..76386100.59 rows=635 width=42)
-> Index Scan using yagoindexobject on yagofacts (cost=0.00..76385370.34 rows=144779 width=42)
Filter: ((((subject)::text <> (object)::text) AND ((predicate)::text = '<linksTo>'::text) AND ((object)::text = '<Morocco>'::text)) OR ((object)::text = '<United_States>'::text))

And the time taken is 138699ms which is way higher than the previous 2 cases **and indicates that in this case indexing is the worst possible optimization plan whereas bitmap scan is the best**

Query 4:

- normal:all on:(sort:external merge) (sequential scan) 238198 warm
- seq scan off: (sort:external merge) (bitmap heap scan) (bitmap index scan predicate) 242230 warm
- seq scan off bitmap off sort off: (index scan objectpredicate) 126152 warm
- seq scan off bitmap off sort off: (index scan object) 154920 warm
- seq scan off bitmap off sort on: (sort:external merge) (index scan predicate) 242511 warm executed if object is removed even if sort is off
- seq scan off index scan off sort off: same as 2
- bitmap scan off index scan off sort off: same as 1

Default Query Plan:

In Query 4 by default seq. scan takes place with foolowing plan:

Query Plan:
GroupAggregate (cost=4710345.92..4840550.06 rows=75627 width=42)
-> Sort (cost=4710345.92..4753495.21 rows=17259716 width=42)
Sort Key: object
-> Seq Scan on yagofacts (cost=0.00..511891.92 rows=17259716 width=42)

Filter: (((subject)::text <> (object)::text) AND ((predicate)::text = '<linksTo>'::text))

And the time taken is 238198ms. Note that here firstly sorting takes place
If seq. scan is turned off then sorting followed by bitmap scan is performed with the following plan:

Query Plan:
GroupAggregate (cost=5087493.51..5217697.65 rows=75627 width=42)
-> Sort (cost=5087493.51..5130642.80 rows=17259716 width=42)
Sort Key: object
-> Bitmap Heap Scan on yagofacts (cost=416808.79..889039.51 rows=17259716 width=42)
Recheck Cond: ((predicate)::text = '<linksTo>'::text)
Filter: ((subject)::text <> (object)::text)
-> Bitmap Index Scan on yagoindexpredicate (cost=0.00..412493.86 rows=17346448 width=0)
Index Cond: ((predicate)::text = '<linksTo>'::text)

And the time taken is242230ms which is similar to first case
But if bitmap scan along with sorting is turned off then only index scan takes place on index objectpredicate(if present) or object with following plan:

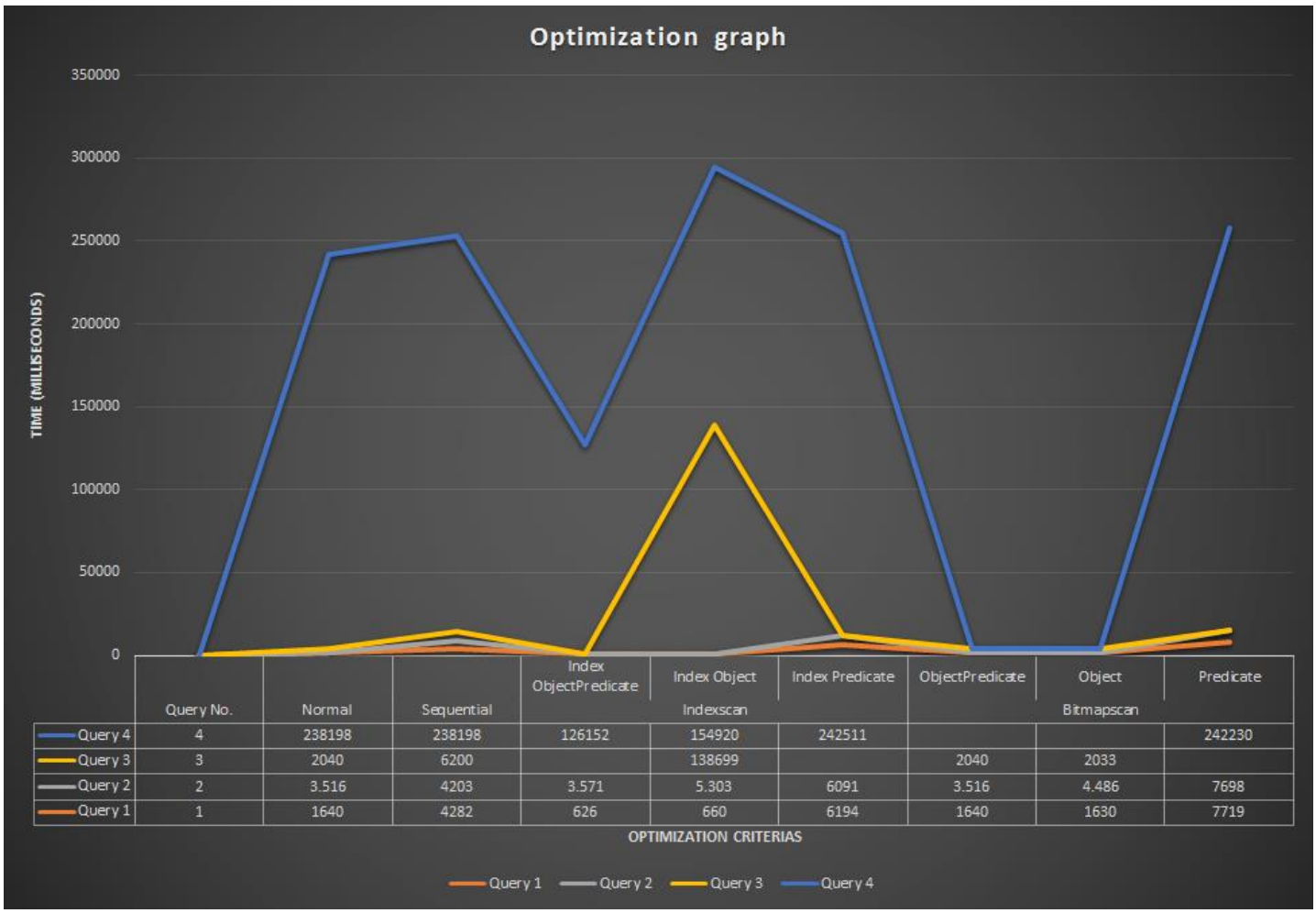
Query Plan:
GroupAggregate (cost=0.00..66937357.87 rows=75627 width=42)
-> Index Scan using yagoindexobjectpredicate on yagofacts (cost=0.00..66850303.02 rows=17259716 width=42)
Index Cond: ((predicate)::text = '<linksTo>'::text)
Filter: ((subject)::text <> (object)::text)

And the time taken is almost half of that taken in previous 2 **cases indicating that index only scan is the best option in this case and rest are almost comparable**

A table and a graph containing: i) Timing of each query without any optimisations (that is, the default database created), ii) Timing of each query with each specifc optimisation that you tried (including, building different kinds of indexes, turning on and of certain algorithms, or a combination of these, etc.). A minimum of 4 different kinds optimisations are expected here. Please ensure that your x-axis is properly labeled!!

Query No.	Normal (ms)	Sequential (ms)	Indexscan (ms)			Bitmapscan (ms)		
			Index ObjectPredicate	Index Object	Index Predicate	Index ObjectPredicate	Index Object	Index Predicate
1	1640	4282	626	660	6194	1640	1630	7719
2	3.516	4203	3.571	5.303	6091	3.516	4.486	7698
3	2040	6200	-	138699	-	2040	2033	-
4	238198*	238198*	126152	154920*	242511*	-	-	242230*

Note= * represents that sorting is also taking place



	Query No.	Normal	Sequential	Index ObjectPredicate	Index Object Indexscan	Index Predicate	ObjectPredicate	Object Bitmapscan	Predicate
Query 4	4	238198	238198	126152	154920	242511			242230
Query 3	3	2040	6200		138699		2040	2033	
Query 2	2	3.516	4203	3.571	5.303	6091	3.516	4.486	7698
Query 1	1	1640	4282	626	660	6194	1640	1630	7719