Cs631 Implementation of Relational Database Systems
Assignment 2(Bitmap Index Scan)

Pranay Dhondi
120050054
KV Gangadhar
120050078

Q1) Successfully loaded the university schema into the local postgres server

Q2) Studied the query plan & time taken for each of the following commands mentioned.

Q3)

1)Show a selection query whose chosen plan uses a bitmap index scan. You can create indices on appropriate relation attributes to create such a case.

A) From given examples query "select * from takes natural join student where ID = '1234';" found to use the bit-map index scan. Above query uses bit-map scan to get all the blocks of Index 'takes_pkey' of "takes" table which are relevant to condition id = '1234'. Later on it scans on the blocks that got by bitmap scan to get the all the relevant tuples(records) which has id='1234'. I think postgres used bitmap inscan instead of normal index scan because selection of a attribute id in relation 'takes' may be evenly distributed i.e, SC(id,'takes) = (cardinality of ('takes') / V(id,'takes)) where V('id','takes') is number of distinct values that appear in the relation 'takes' for a attribute 'id'.

other query is "select * from time_slot natural join section where time_slot.time_slot_id ='A';" also uses bit-map scan.

2) Create a nested subquery with an exists clause which gets decorrelated by PostgreSQL

A) From given examples query "select * from student where exists(select * from instructor where instructor.name = student.name);" is decorrelated by PostgreSQL. If a query is a correlated query(nested subquery), then the inner query refers any attribute of outer query. PostgreSQL evaluates a correlated query by evaluating the subquery separately for each tuple in the outer level query. Due to that a large no of disk I/O may result. Decorrelation involves process of converting a correlated query into an unnested query that gives similar result as the original correlated query gives. Instead for every record in 'student' finding whether the any record in the 'instructor' relation has same name of it's name, postgres converts into a unnested query in which it computes hash aggregate on 'instructor' relation attribute name and then for every record in 'student' calculates hash for its attribute name and checks whether any element is present in the bucket corresponding to the hash value corresponding to its name.

other Query "SELECT * from instructor where exists (select * from department where instructor.dept_name = department.dept_name);" is also decorrelated by PostgreSQL. If a query is a correlated query(nested subquery), then the inner query refers any attribute of outer query. PostgreSQL evaluates a correlated query by evaluating the subquery separately for each tuple in the outer level query. Due to that a large no of disk I/O may result. Decorrelation involves process of converting a correlated query into an unnested query that gives similar result as the original correlated query gives. Instead for every record in 'instructor' finding whether the any record in the ‘department’ relation has same name of it's name, postgres converts into a unnested query in which it computes has hash function on ‘department’ relation attribute dept_name and then for every record in ‘instructor’ calculates hash value for its attribute dept_name and checks whether any element is present in the bucket corresponding to the hash value corresponding to its dept_name.

3)Create a more complex nested subquery, which PostgreSQL is unable to decorrelate. (Hint: use aggregation in the subquery)

A) From given examples query “select * from student where tot_cred = (select sum(credits) from course where course.dept_name = student.dept_name)” is not decorrelated by PSQL.

other examples are “SELECT * from instructor where exists (select count(*) from teaches group by teaches.id HAVING teaches.id = instructor.id);” , “SELECT * from instructor where exists (select department.dept_name,sum(budget) from department group by department.dept_name HAVING department.dept_name = instructor.dept_name);”

4) Create a query which has a subquery in the where clause, where the subquery implements group-by/aggregation. Now add a equality condition on the groupby attribute in the outer query. Find out what plan is chosen by PostgreSQL. Does this match what you expected? Explain your answer.

A) Query is “select * from instructor as A where A.salary = (select avg(salary) from instructor as B group by B.dept_name having A.dept_name = B.dept_name);”.

Plan Executed on postgres is :-

```
------------------------------------------------------------------------
 Seq Scan on instructor a  (cost=0.00..6845.40 rows=2 width=154)
   Filter: (salary = (SubPlan 1))
   SubPlan 1
          ->  GroupAggregate  (cost=0.00..15.52 rows=1 width=72)
          ->  Seq Scan on instructor b  (cost=0.00..15.50 rows=2 width=72)
                    Filter: ((a.dept_name)::text = (dept_name)::text)
(6 rows)
---------------------------------------------------------------------------
```

This goes according to the expected plan, because as we saw in previous example that postgres is unable to decorrelate the complex queries which contain Aggregate operator.