

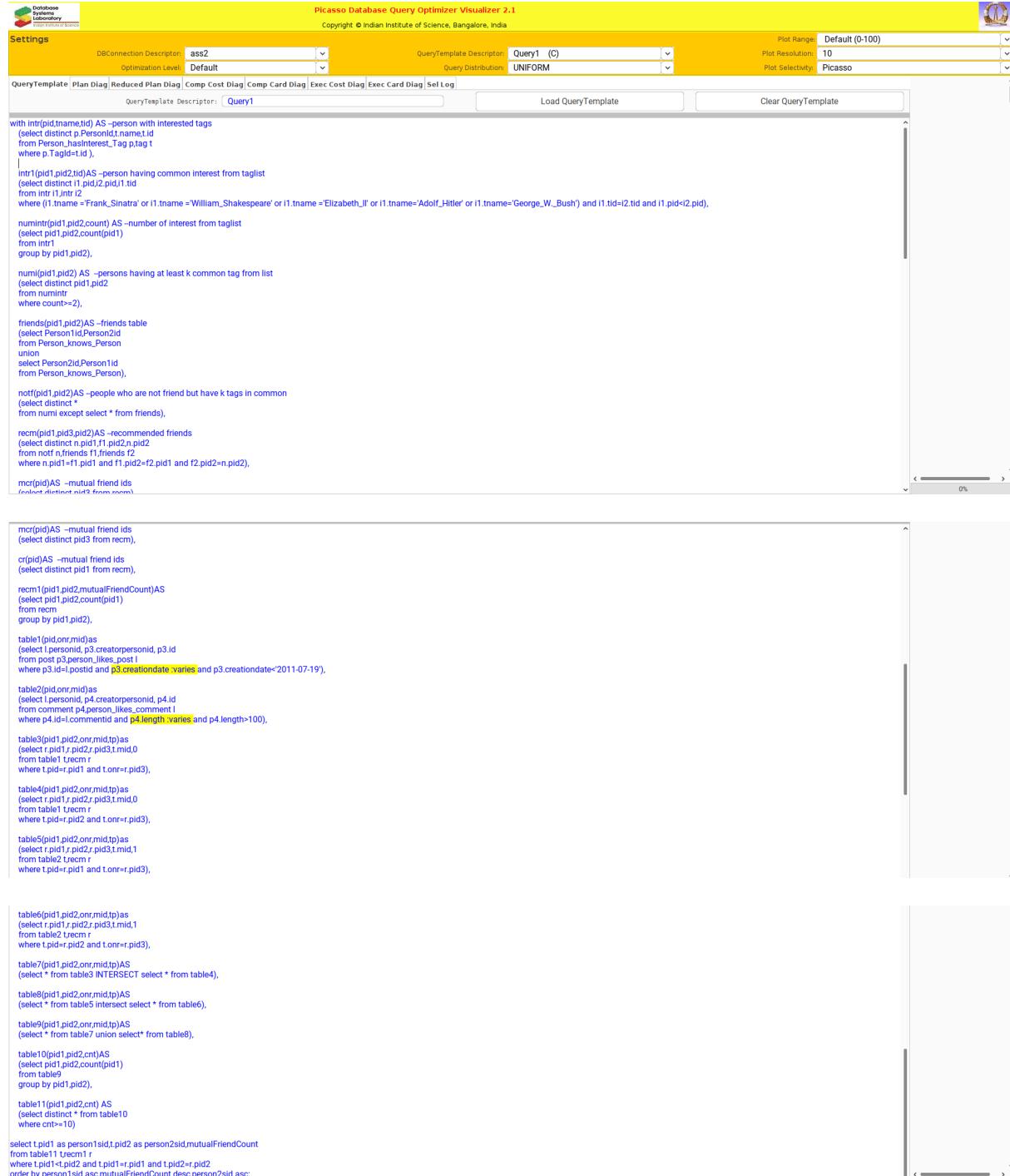
COL362

ASSIGNMENT 2

Sibasish Rout(2020CS10386)
Ankit Raushan (2020CS10324)

1. QUERY1

1.1 Query Template



The screenshot shows the Picasso Database Query Optimizer Visualizer 2.1 interface. The top menu bar includes 'File', 'Edit', 'View', 'Help', and 'About'. The main window has tabs for 'QueryTemplate', 'Plan Diag', 'Reduced Plan Diag', 'Comp Cost Diag', 'Comp Card Diag', 'Exec Cost Diag', 'Exec Card Diag', and 'Sel Log'. The 'QueryTemplate' tab is selected, showing the following SQL query:

```

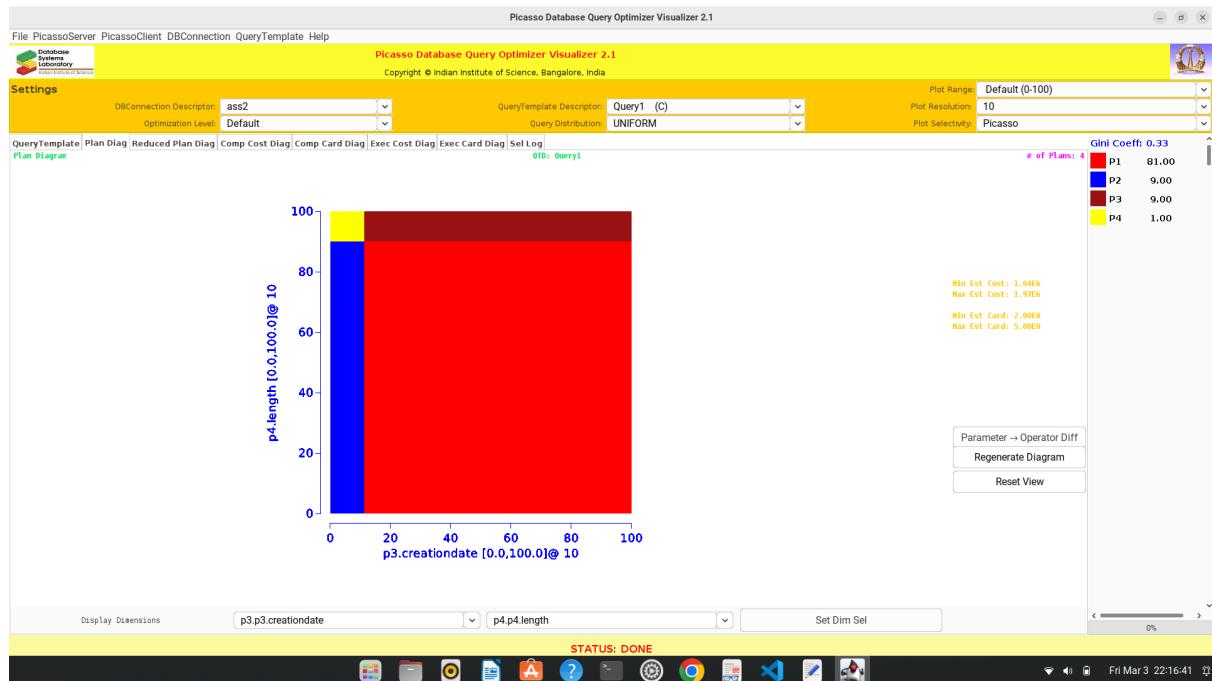
with intr(pid,tname,tid) AS {person with interested tags
  (select distinct p.PersonId,p.Name,p.TagId
   from Person_hasInterest_Tag p,tag t
   where p.TagId=t.Id),
  intr1(pid1,pid2,tid) AS {persons having common interest from taglist
  (select distinct t1.pid,t2.pid,t1.tid
   from intr1 t1,intr1 t2,
   where t1.tname='Frank_Sinatra' or t1.tname='William_Shakespeare' or t1.tname='Elizabeth_Jr' or t1.tname='Adolf_Hitler' or t1.tname='George_W_Bush' and t1.tid=t2.tid and t1.pid<=t2.pid),
  numintr(pid1,pid2,count) AS {number of interest from taglist
  (select pid1,pid2,count(pid1)
   from intr1
   group by pid1,pid2),
  numl(pid1,pid2) AS {persons having at least k common tag from list
  (select distinct pid1,pid2
   from numintr
   where count=>2),
  friends(pid1,pid2) AS {friends table
  (select Person1Id,Person2Id
   from Person_knows_Person
   union
   select Person2Id,Person1Id
   from Person_knows_Person),
  notfp(pid1,pid2) AS {people who are not friend but have k tags in common
  (select distinct *
   from numl
   except select * from friends),
  recm(pid1,pid3,pid2) AS {recommended friends
  (select distinct n.pid1,f1.pid2,n.pid2
   from notfp n,friends f1,friends f2,
   where n.pid1=f1.pid1 and f1.pid2=f2.pid1 and f2.pid2=n.pid2),
  mcr(pid) AS {mutual friend ids
  (select distinct pid3
   from recm),
  mcr(pid) AS {mutual friend ids
  (select distinct pid3
   from recm),
  cr(pid) AS {mutual friend ids
  (select distinct pid1
   from recm),
  recm1(pid1,pid2,mutualFriendCount) AS
  (select pid1,pid2,count(pid1)
   from recm
   group by pid1,pid2),
  table1(pid,tn,mid) AS
  (select t.PersonId, p3.CreatorPersonId, p3.Id
   from Post p3,Person t
   where p3.Id=t.PostId and p3.CreationDate > '2011-07-19'),
  table2(pid,tn,mid) AS
  (select t.PersonId, p4.CreatorPersonId, p4.Id
   from Comment p4,Person t
   where p4.Id=t.CommentId and p4.Length > 100),
  table3(pid1,pid2,tn,mid,tp) AS
  (select r.pid1,r.pid2,r.pid3,t.mid,0
   from table1 t,recm r
   where r.pid1=pid1 and t.onr=r.pid3),
  table4(pid1,pid2,tn,mid,tp) AS
  (select r.pid1,r.pid2,r.pid3,t.mid,0
   from table1 t,recm r
   where r.pid1=pid2 and t.onr=r.pid3),
  table5(pid1,pid2,tn,mid,tp) AS
  (select r.pid1,r.pid2,r.pid3,t.mid,1
   from table2 t,recm r
   where r.pid1=pid1 and t.onr=r.pid3),
  table6(pid1,pid2,tn,mid,tp) AS
  (select r.pid1,r.pid2,r.pid3,t.mid,1
   from table2 t,recm r
   where r.pid1=pid2 and t.onr=r.pid3),
  table7(pid1,pid2,tn,mid,tp) AS
  (select * from table3 INTERSECT select * from table4),
  table8(pid1,pid2,tn,mid,tp) AS
  (select * from table5 intersect select * from table6),
  table9(pid1,pid2,tn,mid,tp) AS
  (select * from table7 union select* from table8),
  table10(pid1,pid2,tn) AS
  (select pid1,pid2,count(pid1)
   from table9
   group by pid1,pid2),
  table11(pid1,pid2,tn) AS
  (select distinct t.pid1 as person1sid,t.pid2 as person2sid,mutableFriendCount
   from table11 t,recm1 r
   where t.pid1<=pid2 and t.pid1>=pid1 and t.pid2>=pid1
   order by person1sid asc,mutableFriendCount desc,person2sid asc);

```

1.2 Execution Plan Diagram

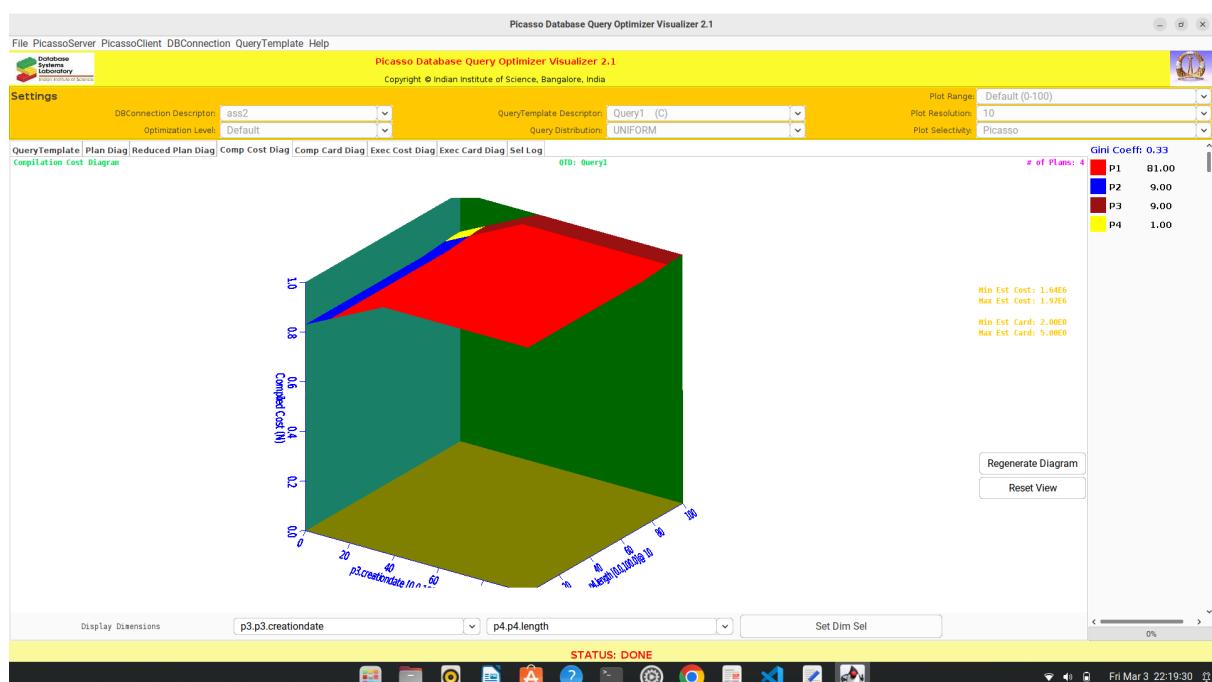
This diagram represents the plan that is optimal at various percentages of the varying tables

For eg here 3 plans are generated for different values of range of length and creationdate
 Which means that if we take 50% of creationdate column and 50% of length column then the Plan P1 is the most optimal as shown by the red value. Similarly we can also deduce the optimal plans for other regions also. The presence of multiple plans implies that we can still Optimise the query for cases when either of the tables has a lower number of values.



1.3 Compilation Cost diagram

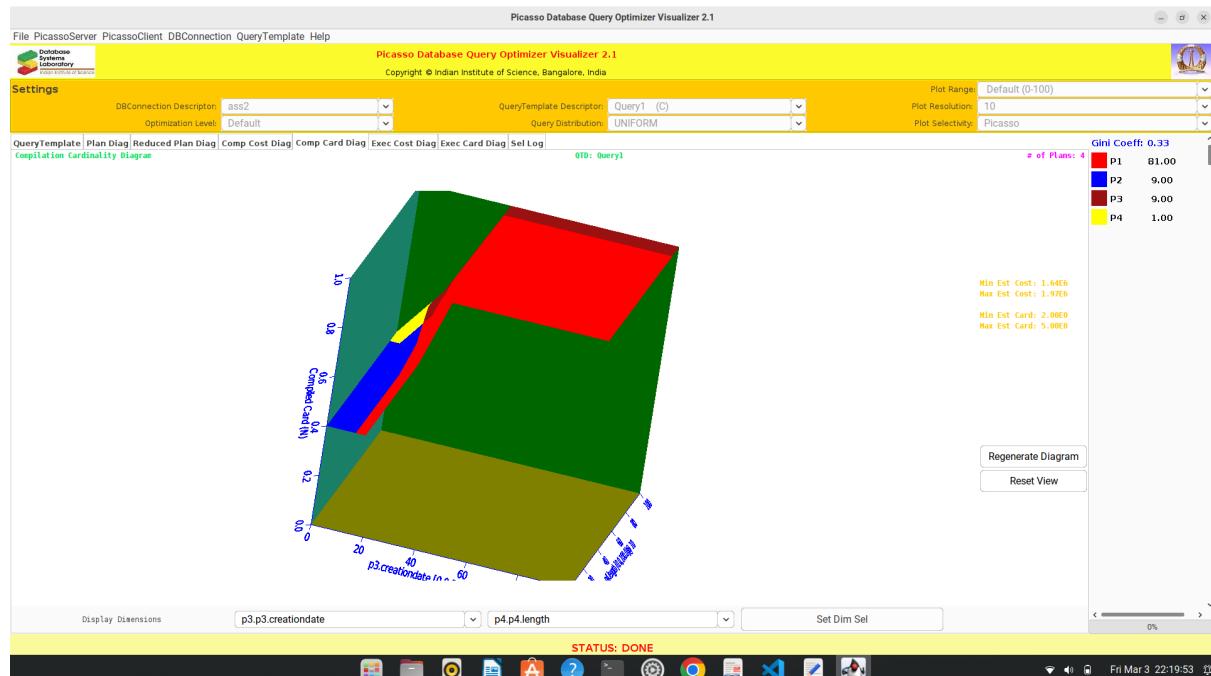
This graph represents the computation time along varying range of length and creationdate along the optimal plan. The value along the z axis in the diagram represents the computation time. The maximum computation time is when we use the whole range for both length and Creationdate. It shows strong correlation with creationdate attribute.



1.4 Compilation Cardinality Diagram

This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1.

From the above diagram we can see that the number of distinct values in the query result Increases with the range of values taken from creationdate .



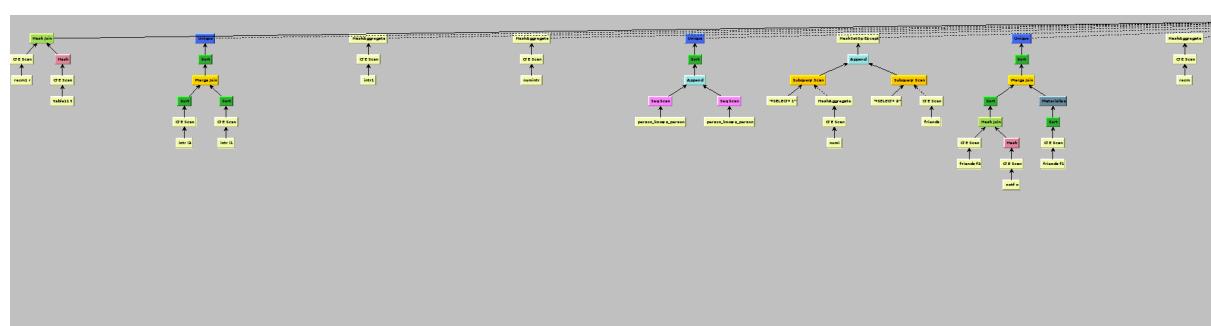
1.5 Plan diagrams

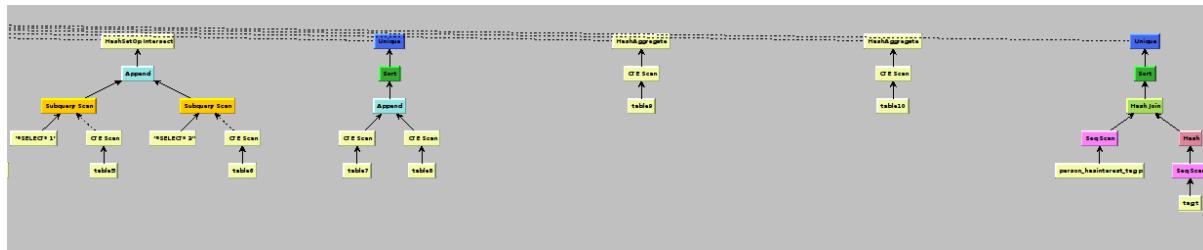
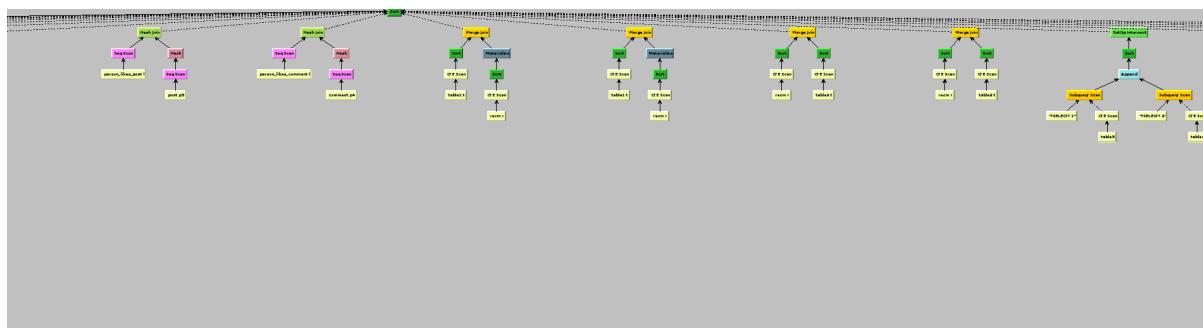
Plan 1



Fig: Total plan 1 diagram

Since plan diagram is not visible I am pasting screenshots dividing it in 3 parts





Plan diagram is overall almost balanced and distributed. From the diagram we can see that Merge join, hash join, sort, append, scan etc. has been used.

Like table1 and table2 are hashed separately and then merge joined,

table5 and table6 is scanned separately and then appended,

table7 and table 8 have been appended,

Person_hasinterest_tag is scanned tag table is hashed and both are then hash joined

Plan 2

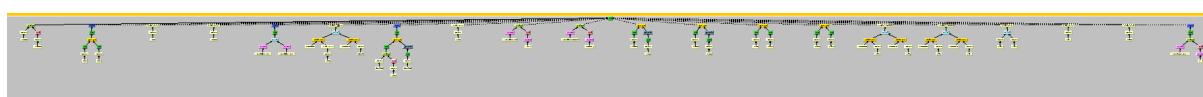
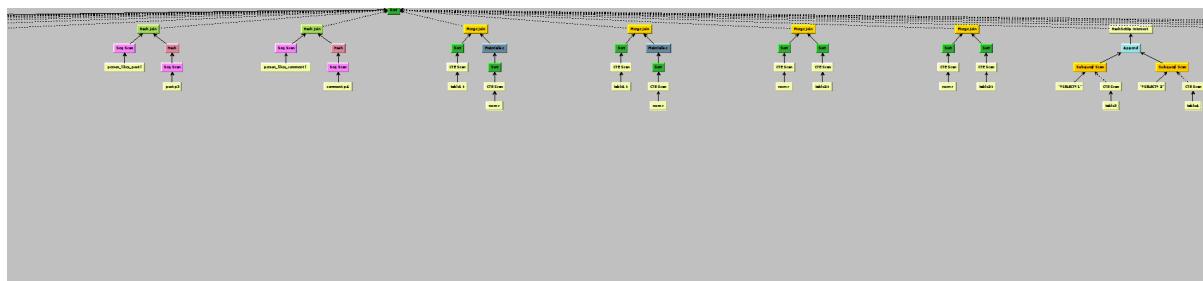
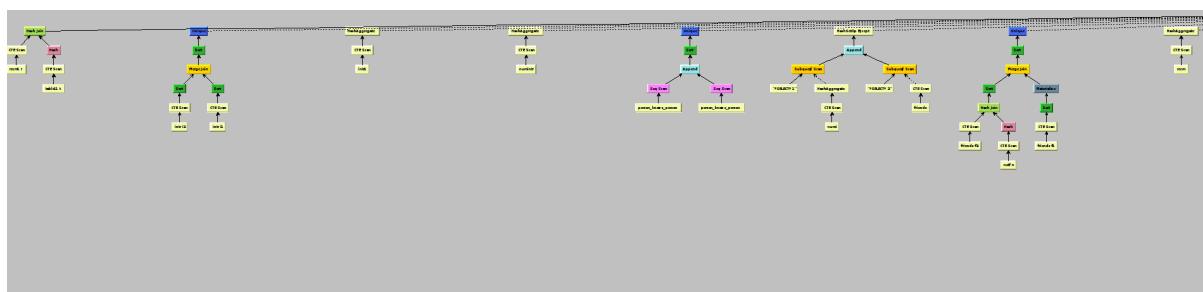
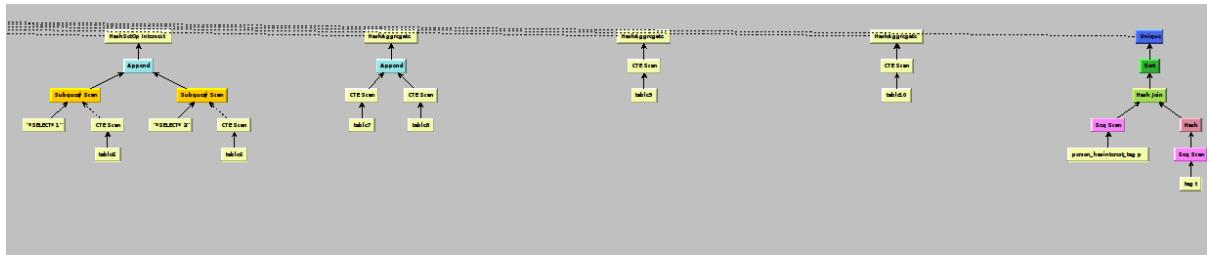


Fig: Total plan 2 diagram

Screenshots of splitted plan 2 diagram





Plan diagram is overall almost balanced and distributed like plan 1

Merge join, hash join, sort, append, scan etc. has been used.

Plan 3

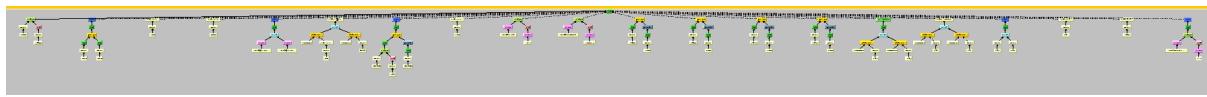
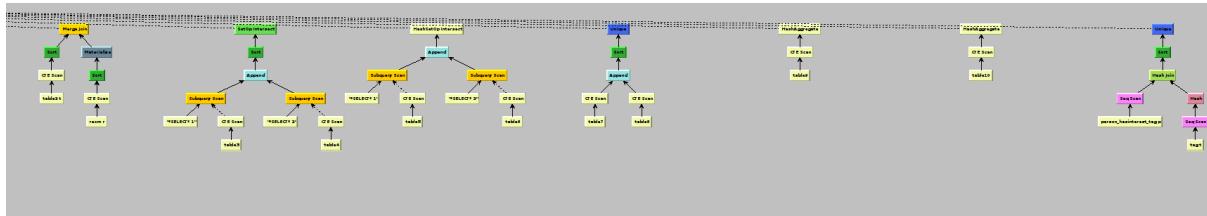
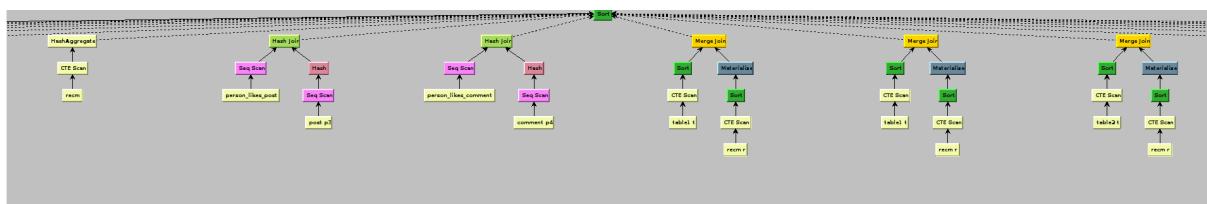
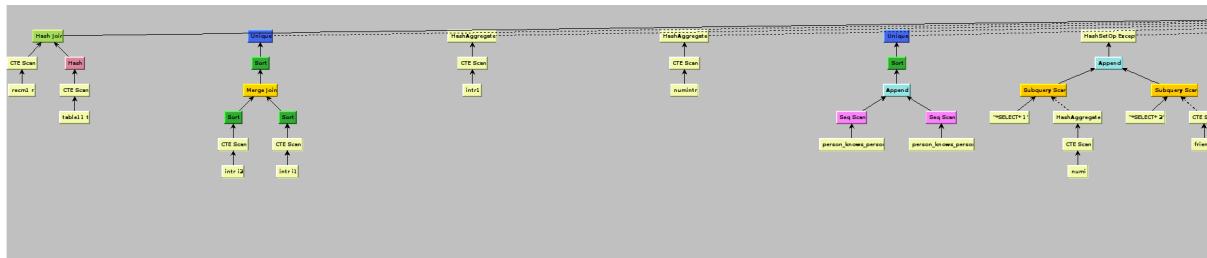


Fig: Total plan 3 diagram

Screenshots of plan 3 after splitting it



Plan diagram is overall almost balanced and distributed like plan 1 and 2

Merge join, hash join, sort, append, scan etc. has been used.

Plan 4

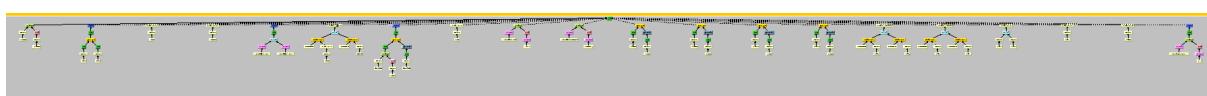


Fig: Total plan 4 diagram

Screenshot of splitted plan 4



Plan diagram is overall almost balanced and distributed like plan 1,2 and 3

Merge join, hash join, sort, append etc. has been used.

2. QUERY 2

2.1 Query Template

Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

Settings

- DB Connection Descriptor: ass2
- QueryTemplate Descriptor: Query2 (C)
- Plot Range: Default (0-100)
- Optimization Level: Default
- Query Distribution: UNIFORM
- Plot Resolution: 10
- Plot Selectivity: Picasso

QueryTemplate Descriptor: [Query2](#) [Load QueryTemplate](#) [Clear QueryTemplate](#)

```

with
tab0(cid,cname)AS
(
  select t1.id,t2.name
  from place t1,place t2
  where t1.type='City' and t1.PartOfPlaceId=t2.id
),
table0(id,birthday)AS
  (select p1.id,p1.birthday
  from person p1,tab0 t1
  where p1.creationDate>'2010-06-01' and p1.creationDate_varies and p1.creationDate<'2012-07-01' and p1.LocationCityId=t1.cid and t1 cname='China'),
table1(id,uni)AS--people with same birthmonth
  (select p1.id,p2.id
  from table0,p1.table0 p2
  where EXTRACT(MONTH FROM p1.birthday)=EXTRACT(MONTH FROM p2.birthday)),
table2(id,uni)AS--people and their university
  (select p1.id,p1.UniversityId
  from person p1,table1,p2
  where p1.id=u PersonId and p2.creationDate_varies),
table3(p1.id,p2.id)AS--with same university
  (select p1.id,p2.id
  from table2,p1.table2 p2
  where p1.univ=p2.univ),
table4(p1.id,p2.id)AS--people having same college same birthmonth and account within a time span
  (select p1.id,p2.id
  from table1 intersect * from table3),
table5(p1.id,p2.id)AS
  (select p1.id,p2.id
  from table4,p1.table4 p2
  where p1.college=p2.college),
table6(p1.id,p2.id)AS--people from table4 who are also friend
  (select p1.id,p2.id
  from table4,intersect select * from table5),
table7(p1.id,p2.id)AS
  (select p1.id,p2.id
  from table6),
table8(p1.id,p2.id)AS--(p1,p2) and (p2,p1) both present in this table
  (select *
  from table6 union all select * from table7),
table9(p1.id,p2.id,p3.id)AS
  (select t1.pid1,t1.pid2,t2.pid3
  from table8 t1,table8 t2
  where t1.pid2=t2.pid1 and t1.pid1=t2.pid2 and t2.pid3=t1.pid1
  select count(pid1)/6 as count
  from table9);

```

2.2 Execution plan diagram

This diagram represents the plan that is optimal at various percentages of the varying tables

For eg here 5 plans are generated for different values of range of length and creationdate

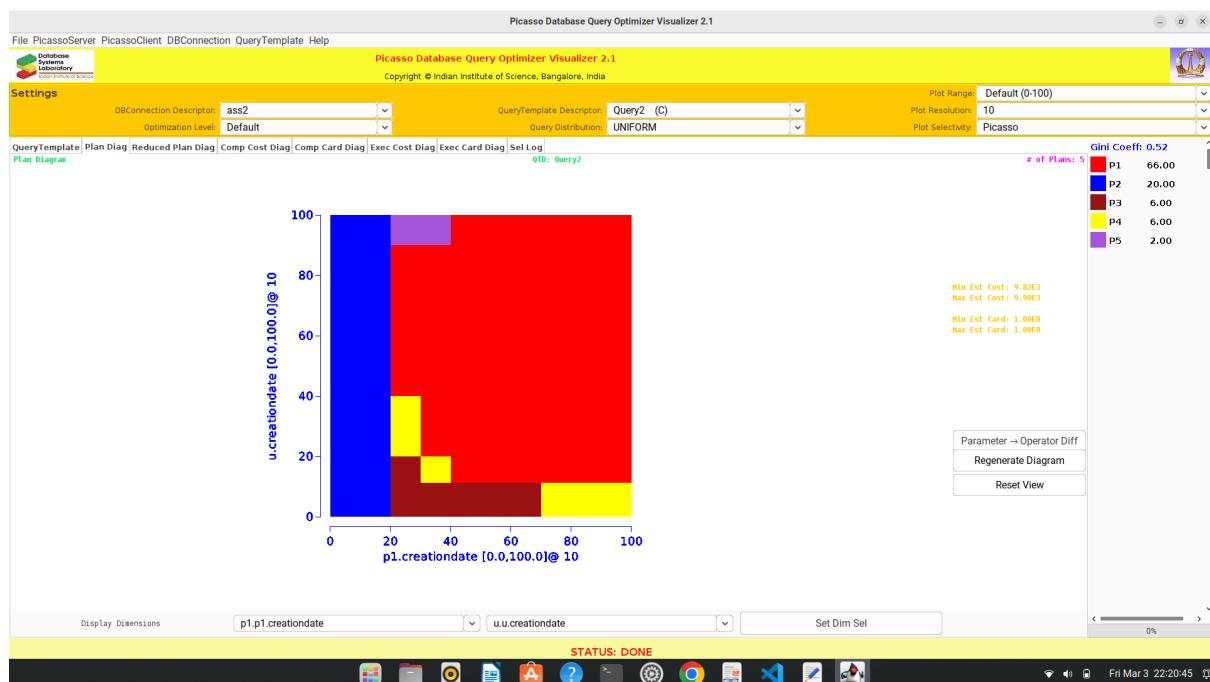
Which means that if we take 50% of person creationdate column and 50% of

Person_studyat_university creationdate column then the

Plan P1 is the most optimal as shown by the red value. Similarly we can also deduce the

optimal plans for other regions also.The presence of multiple plans implies that we can still

Optimise the query for cases when either of the tables has a lower number of values.



2.3 Compilation cost diagram

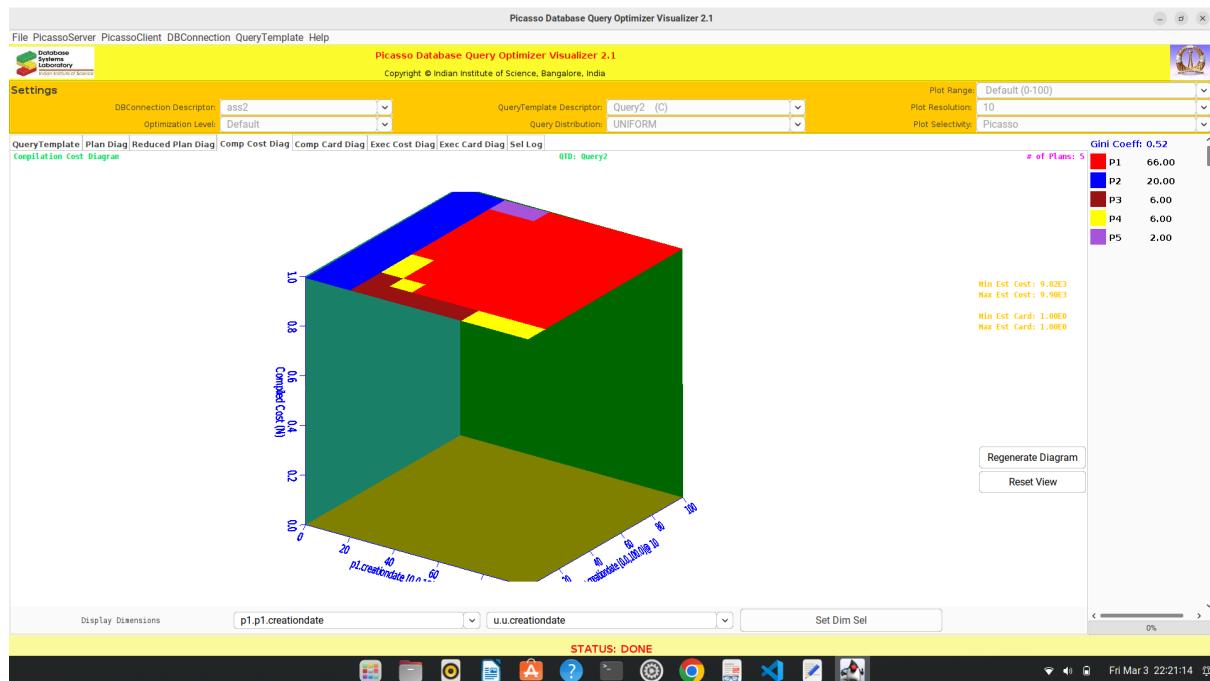
This graph represents the computation time along varying range of person

creationdate and person_studyat_university creationdate

along the optimal plan.The value along the z axis in the diagram represents the computation

time.The maximum computation time is when we use the whole range for both

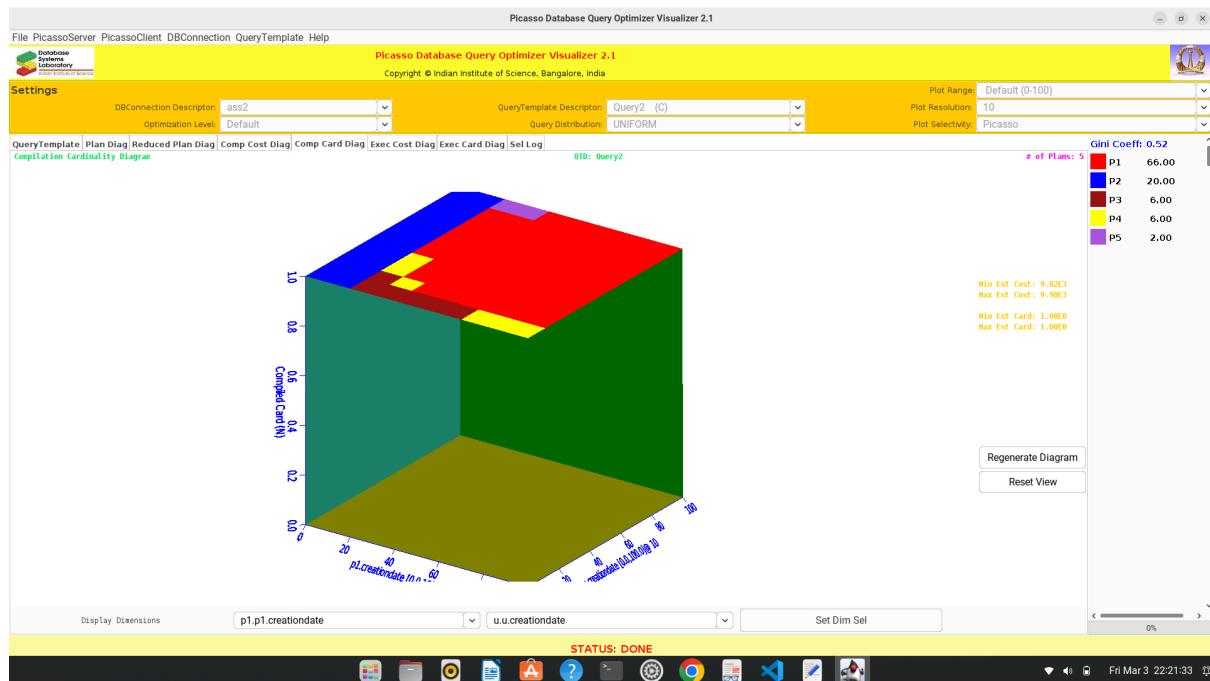
Creationdates.The computation time mostly remains same with change in the predicates.



2.4 Compilation cardinality diagram

This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1.

From the above diagram we can see that the number of distinct values in the query result More or less remains the same with change in predicate values.



2.5 Plan diagrams

Plan 1

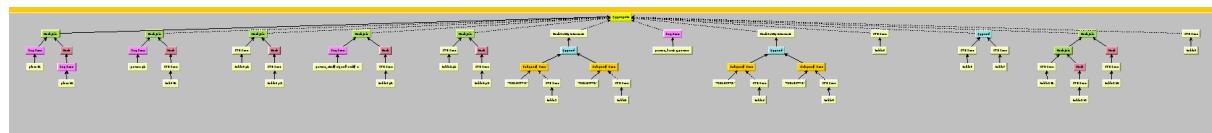
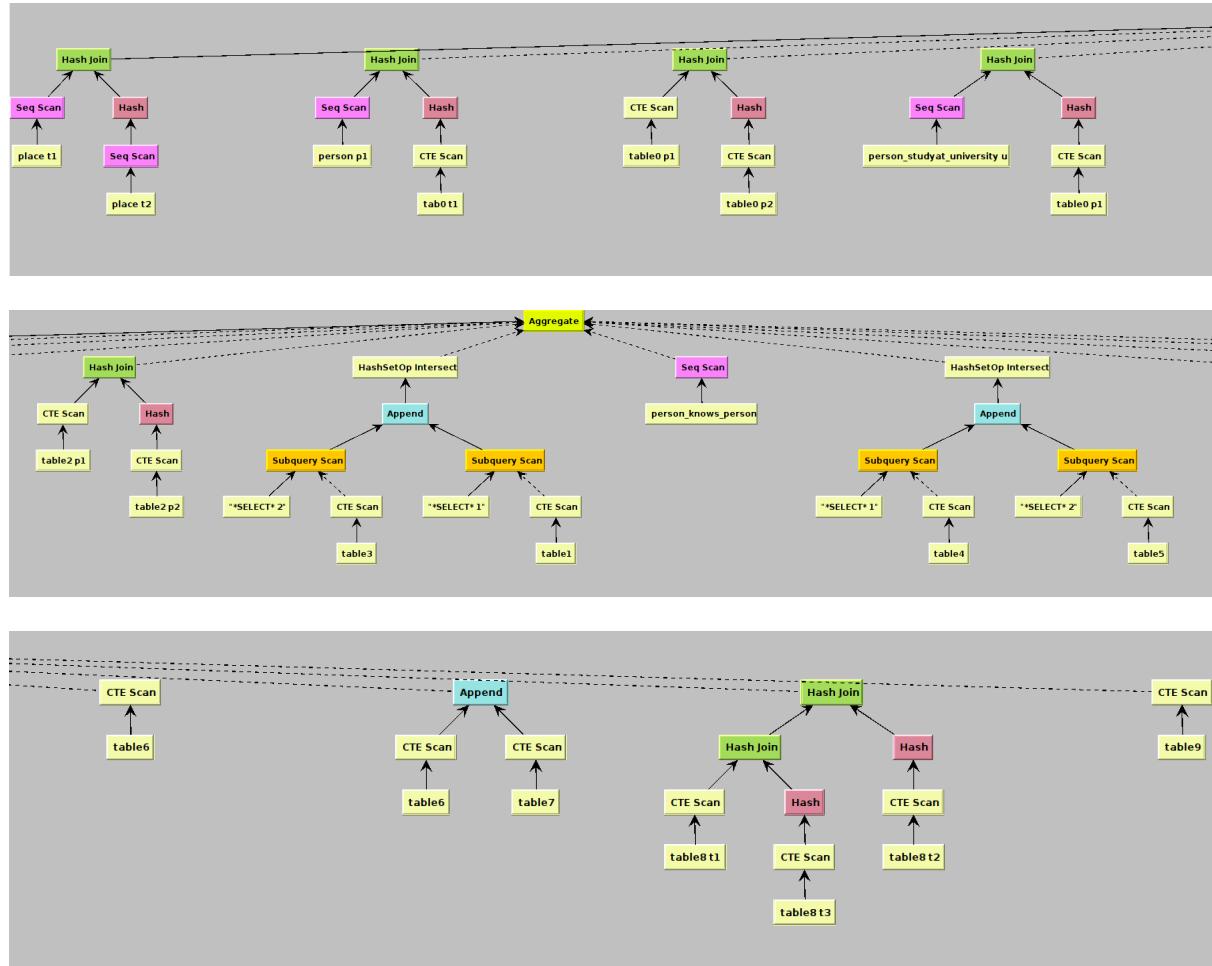


Fig: Total plan 1 diagram

Since it is not clearly visible I am pasting plan diagram by dividing it



From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table,

Table 3 and table 1 are scanned and appended separately,

Table8 t1 is scanned and hash joined with table8 t3, table8 t2 is hashed and then hash Joined with resultant of t1 and t3

Plan 2

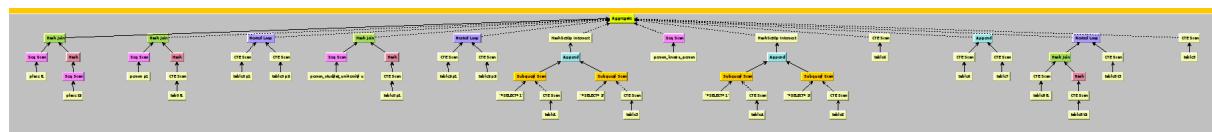
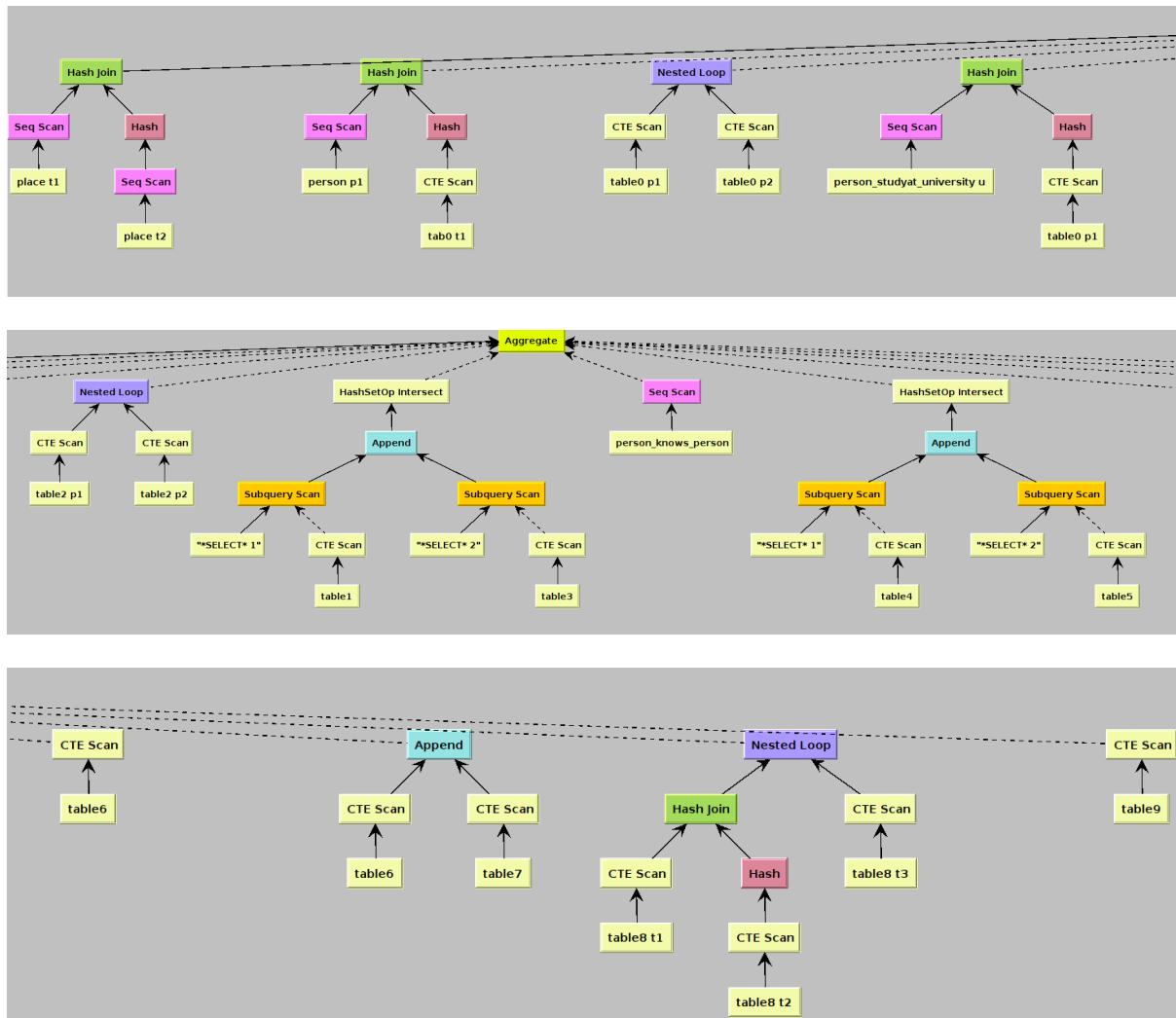


Fig: Total plan 2 diagram



From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table like plan1

Table 3 and table 1 are scanned and appended separately,

Table8 t1 is scanned and hash joined with table8 t2, table8 t3 is hashed and then hash

Joined with resultant of t1 and t2 unlike plan1

Plan 3

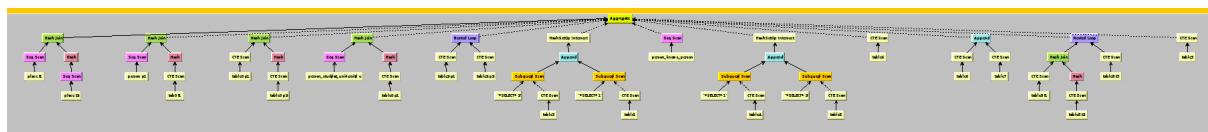
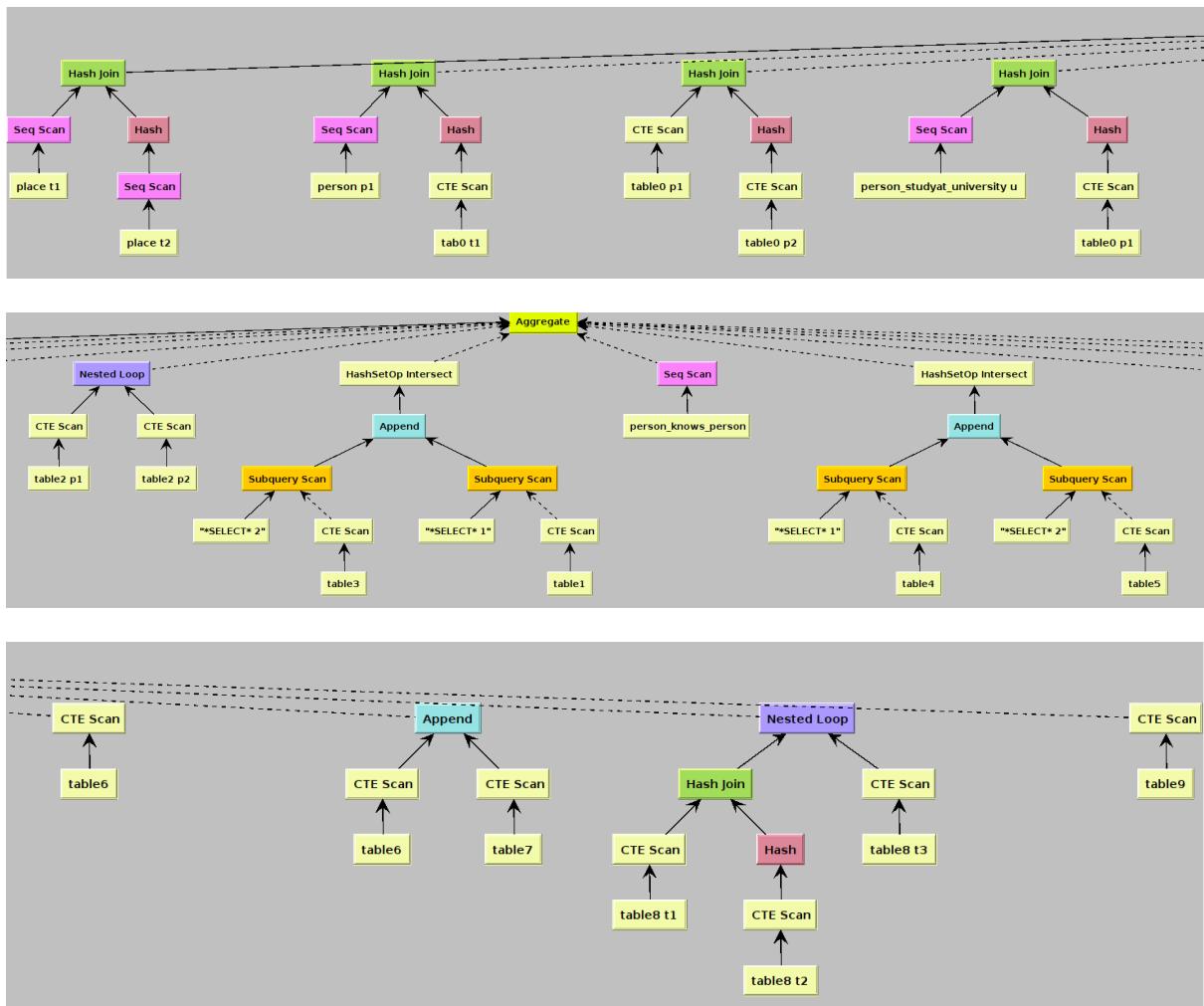


Fig: Total plan 3 diagram



From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table,

Table 3 and table 1 are scanned and appended separately,

Table8 t1 is scanned and hash joined with table8 t2, table8 t3 is hashed and then hash Joined with resultant of t1 and t2

Plan 4

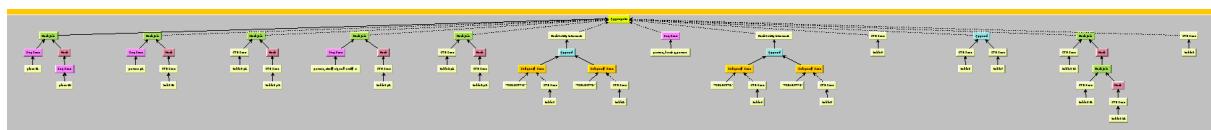
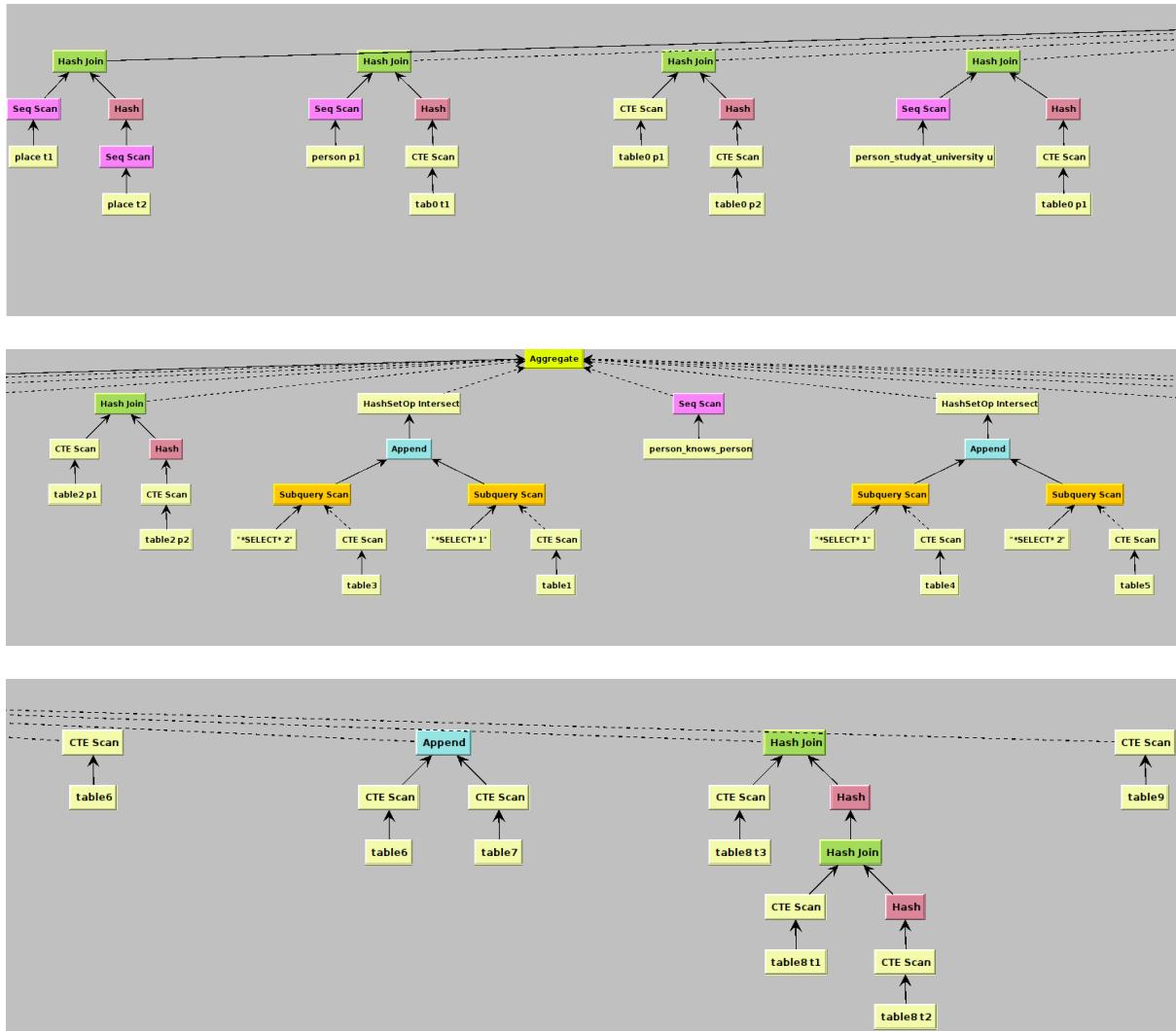


Fig: Total plan 4 diagram



From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table,

Table 3 and table 1 are scanned and appended separately,

Table8 t3 is scanned and hash joined with table8 t1, table8 t2 is hashed and then hash Joined with resultant of t3 and t1

Plan 5

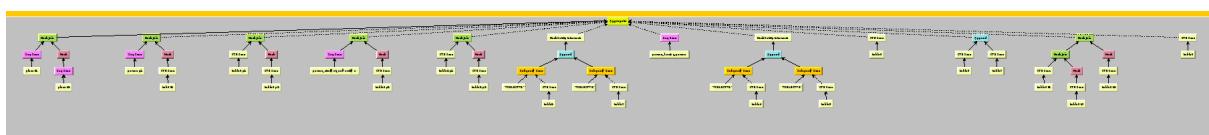
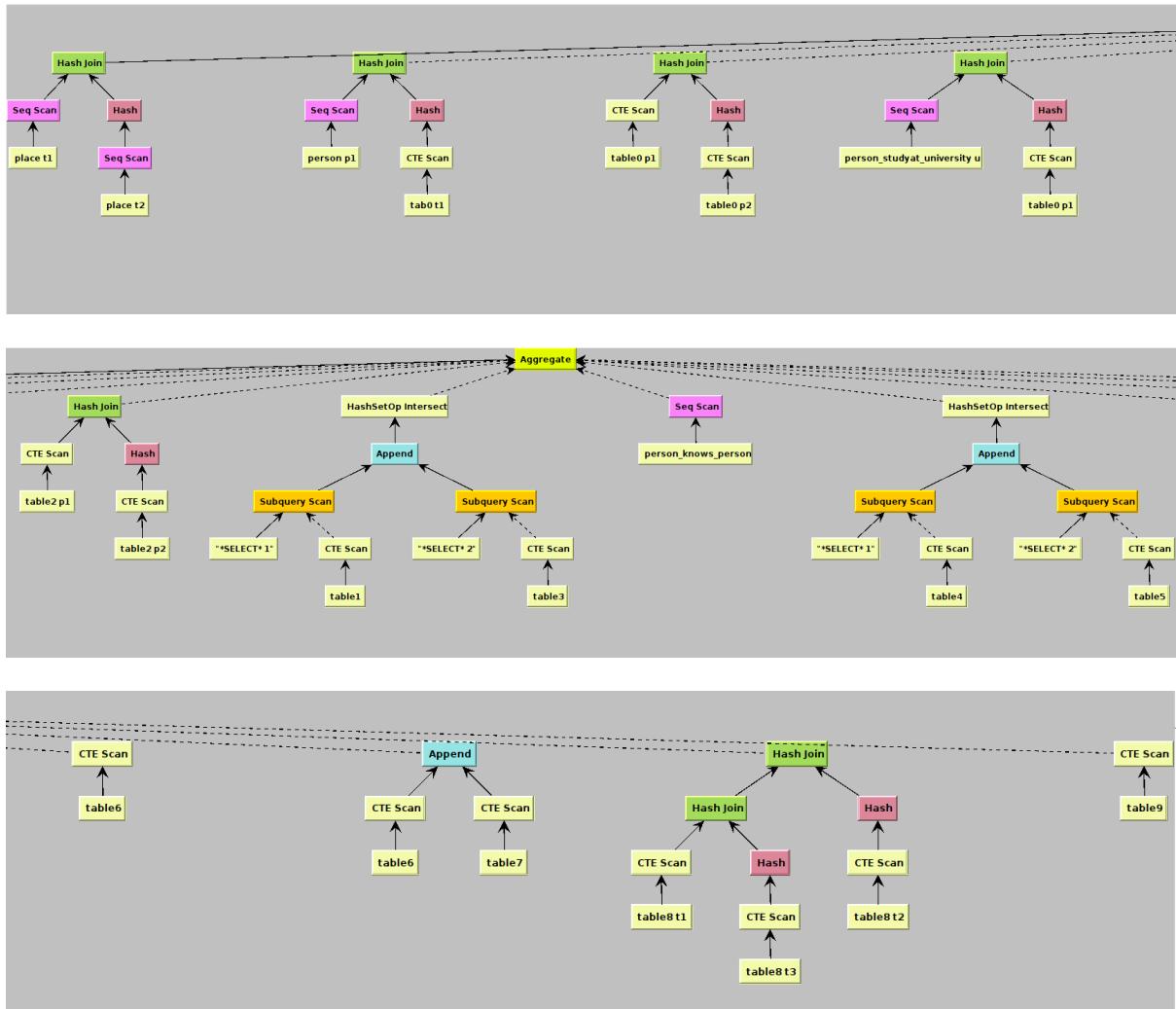


Fig: Total plan 5 diagram



From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table,

Table 3 and table 1 are scanned and appended separately,

Table8 t1 is scanned and hash joined with table8 t3, table8 t2 is hashed and then hash Joined with resultant of t1 and t3

3. QUERY 3

3.1 Query Template

```

with temp1(tid) as
(
  select c.Tagid
  from comment_hashtag_tag c
  where c.creationdate >= '2010-02-03' and c.creationdate <= '2010-12-03'
)
union all
(
  select p.Tagid
  from post_hashtag_tag p
  where p.creationdate >= '2010-02-03' and p.creationDate <= '2010-12-03'
),
temp2(tid) as
(
  select c.Tagid
  from comment_hashtag_tag c
  where c.creationDate >= '2010-12-03' and c.creationDate <= '2011-05-03'
)
union all
(
  select c.Tagid
  from post_hashtag_tag c
  where c.creationDate >= '2010-12-03' and c.creationDate <= '2011-05-03'
),
count1(tid,cmsg) as
(
  select tid, count(*) from temp1 group by tid
),
count2(tid,cmsg) as
(
  select tid, count(*) from temp2 group by tid
),
select tname as tagclassname, count(*) as count from final group by tname order by count desc, tagclassname asc;

```

3.2 Execution Plan Diagram

This diagram represents the plan that is optimal at various percentages of the varying tables

For eg here 1 plan is generated for different values of range of length and creationdate

Which means that for any value of range of post creationdate and comment creationdate

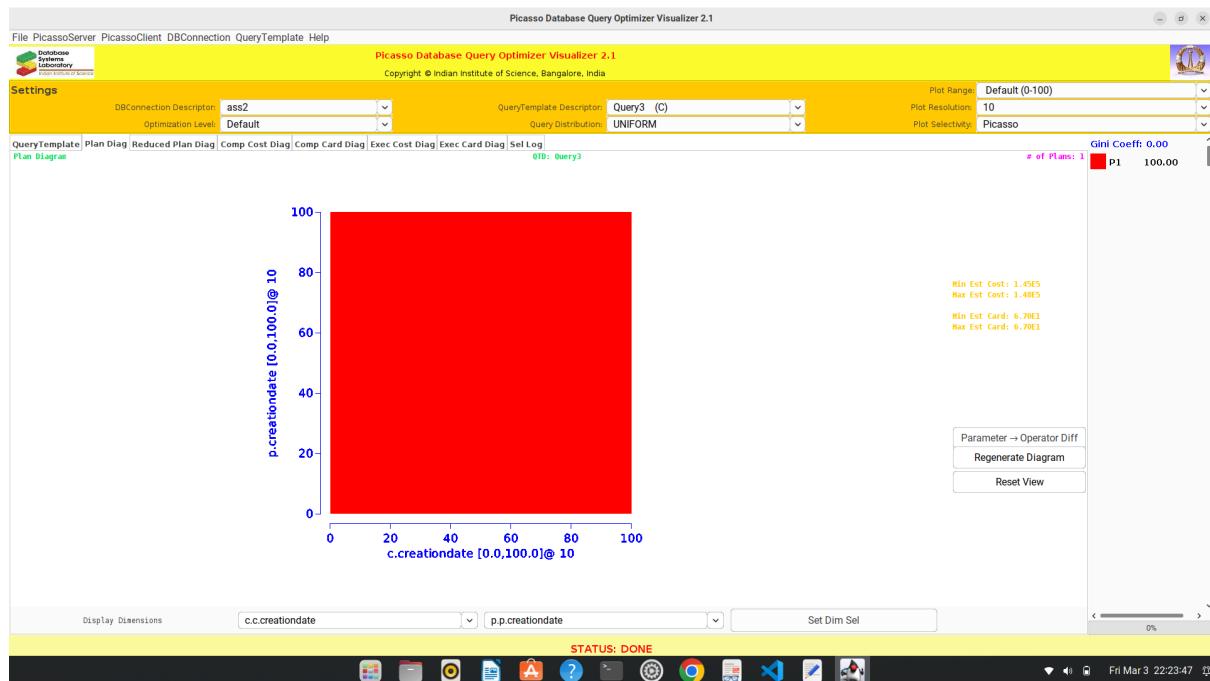
We only have one plan given in red.

.Similarly we can also deduce the optimal plans for other regions also.

The presence of single plan implies that we have a

Optimal query plan for all ranges of predicate values which means that our query is quite

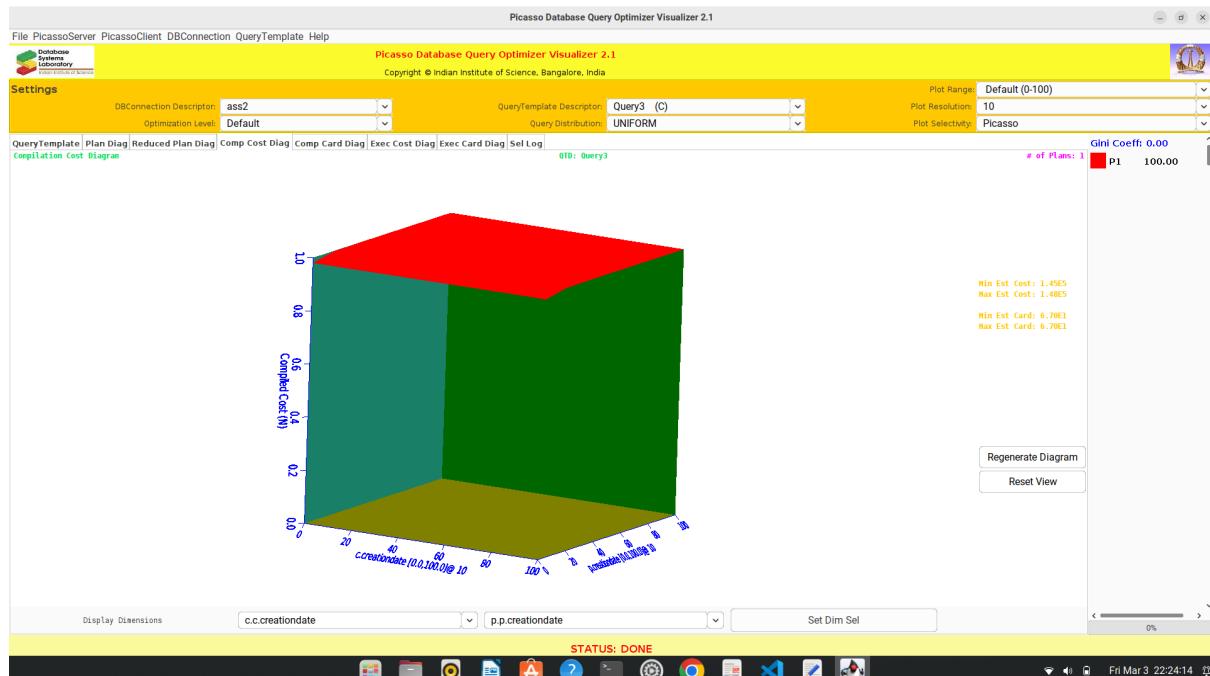
efficient.



3.3 Compilation Cost diagram

This graph represents the computation time along varying range of post creationdate and comment creationdate along the optimal plan. The value along the z axis in the diagram represents the computation time. The maximum computation time is when we use the whole range for creationdate Value of both the predicates

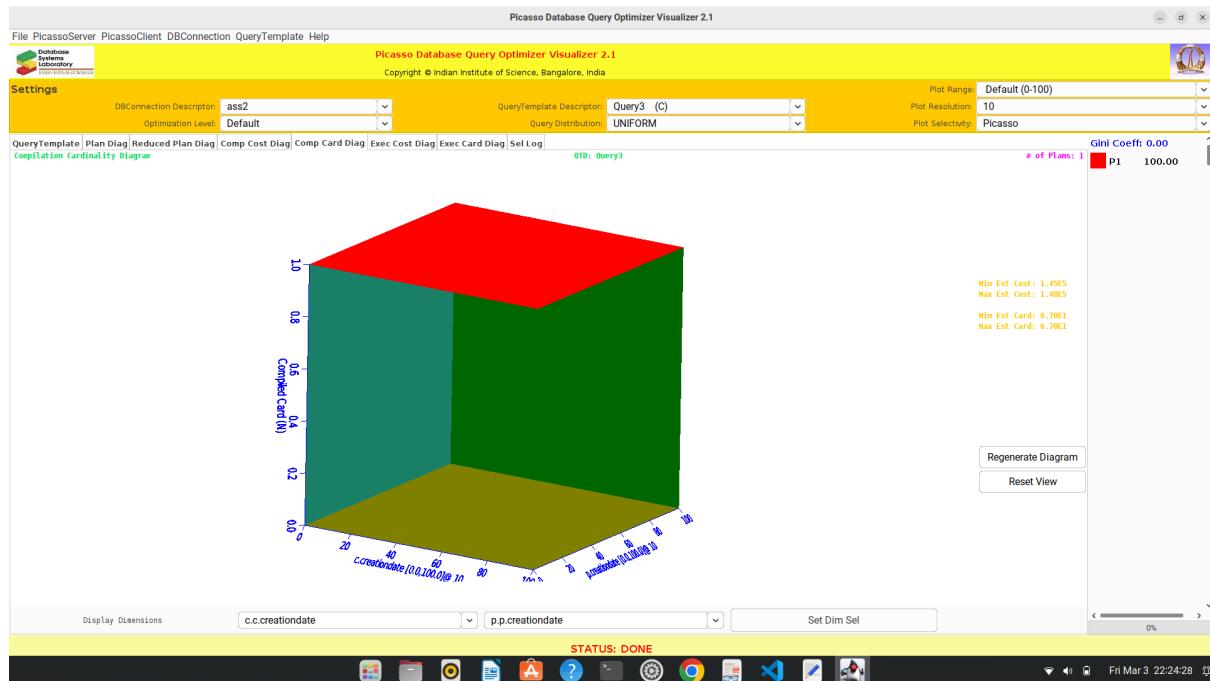
.The computation time mostly remains same with change in the predicates.



3.4 Compilation Cardinality Diagram

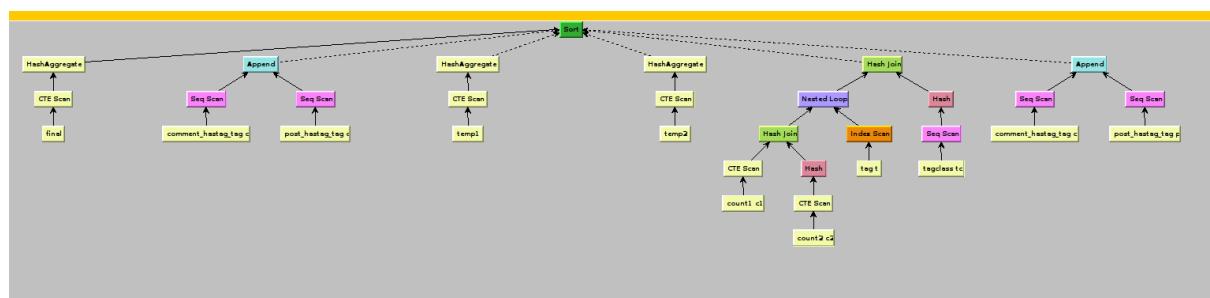
This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1.

From the above diagram we can see that the number of distinct values in the query result More or less remains the same with change in predicate values.



3.5 Plan Diagram

Plan



The plan diagram is quite balanced except along one subtree where it becomes a little Skewed as the height is relatively same across the different Subtrees. The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan in one subtree .

4. QUERY 4

4.1 Query Template

Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

Settings

DBConnection Description: ass2 QueryTemplate Description: **Query 4 (C,E)** Plot Range: Default (0-100)
 Optimization Level: Default Query Distribution: UNIFORM Plot Resolution: 10
 Plot Selectivity: Picasso

QueryTemplate| Plan Diag| Reduced Plan Diag| Comp Cost Diag| Comp Card Diag| Exec Cost Diag| Exec Card Diag| Set Log| Load QueryTemplate| Clear QueryTemplate

```

with post1(pid,cpid) as
(
  select p.id,c.id from post p,comment c
  where c.length >=4 and p.creationdate >=4 and p.id=c.ParentPostId
),
comment1(pid,cpid) as
(
  select c1.id,c2.id from comment c1,comment c2
  where c1.id=c2.ParentCommentId
),
cmsg(pid,cmsg) as
(
  select pid,count(*) from comment1 group by pid
),
pmmsg(pid,cmsg) as
(
  select pid,count(*) from post1 group by pid
),
tcmmsg(tid) as
(
  select cht.TagId from comment_hashtag_tag cht,cmsg
  where c.pid=cht.CommentId and c.cmsg>=4
),
tpmsg(tid) as
(
  select pht.TagId from post_hashtag_tag ph,pmsg p
  where p.pid=pht.PostId and p.cmsg>=4
),
final(tid) as
(
  (select * from tcmmsg) union all (select * from tpmsg)
),
fcount(tid,c) as
(
  select tid,count(*) from final group by tid
)

select t.name as tagname,f.c as count from fcount fc,tag t
where t.id=fc.tid
order by fc.c desc,t.name asc limit 10;
  
```

STATUS: DONE

4.2 Execution Plan Diagram

This diagram represents the plan that is optimal at various percentages of the varying tables

For eg here 1 plan is generated for different values of range of length and creationdate

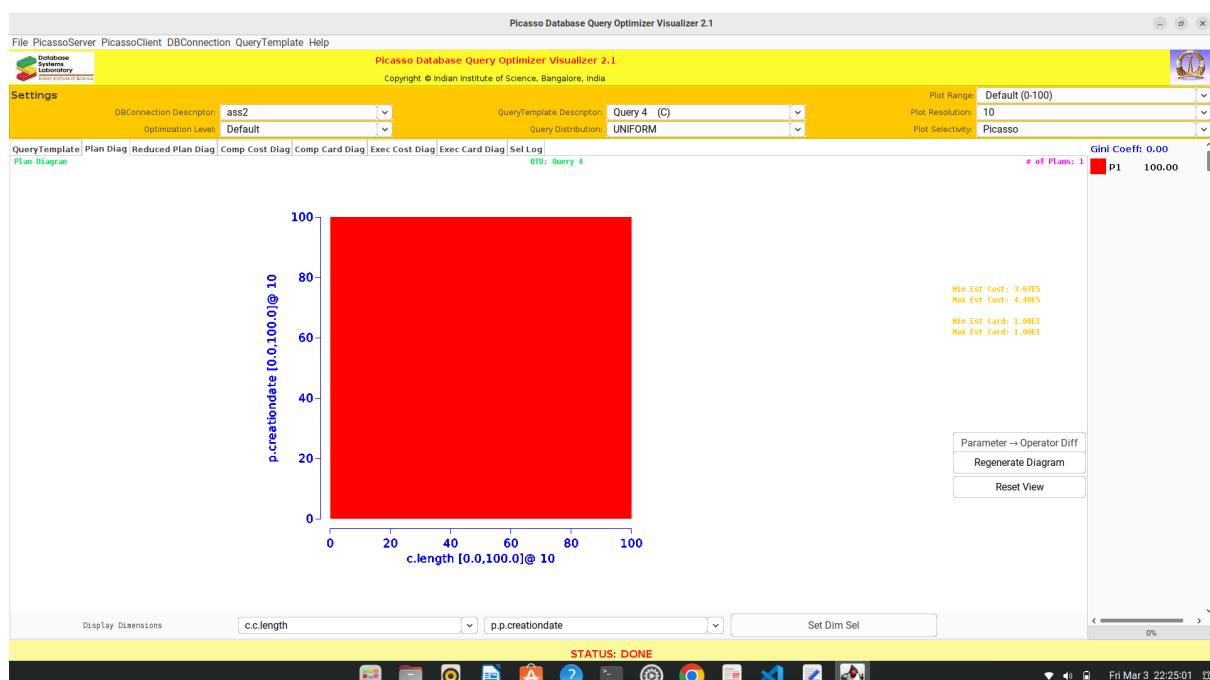
Which means that for any value of range of post creationdate and comment creationdate

We only have one plan given in red.

.Similarly we can also deduce the optimal plans for other regions also.

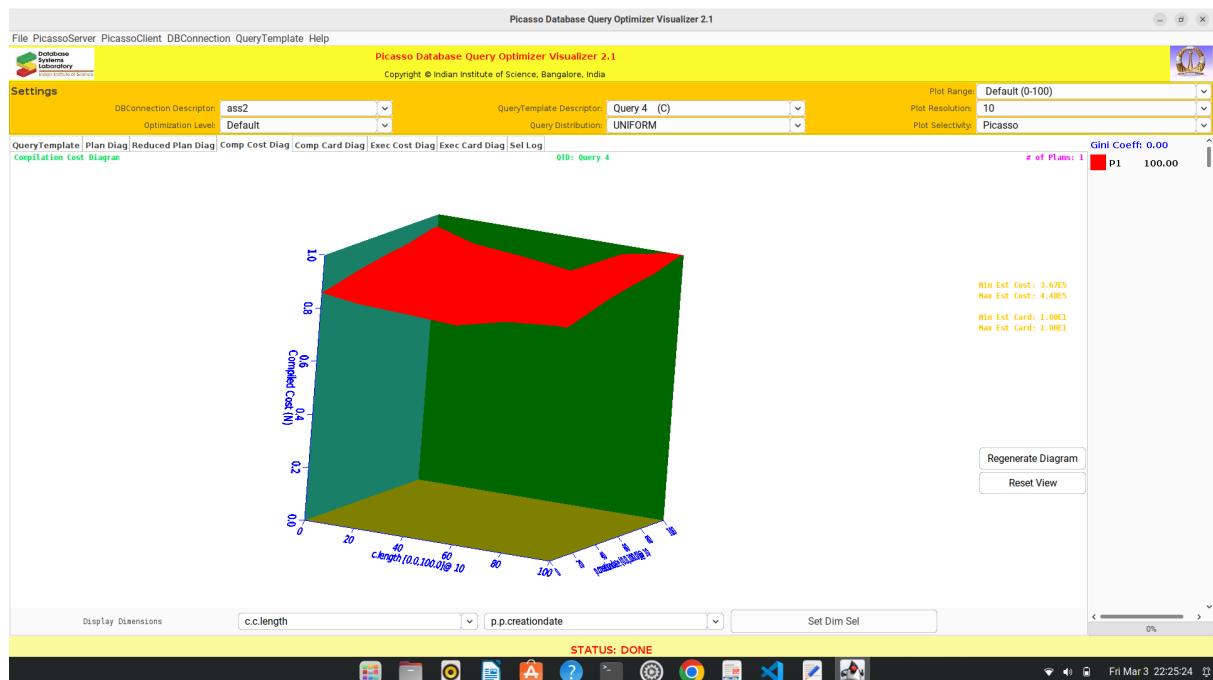
The presence of single plan implies that we have a

Optimal query plan for all ranges of predicate values which means that our query is quite efficient.



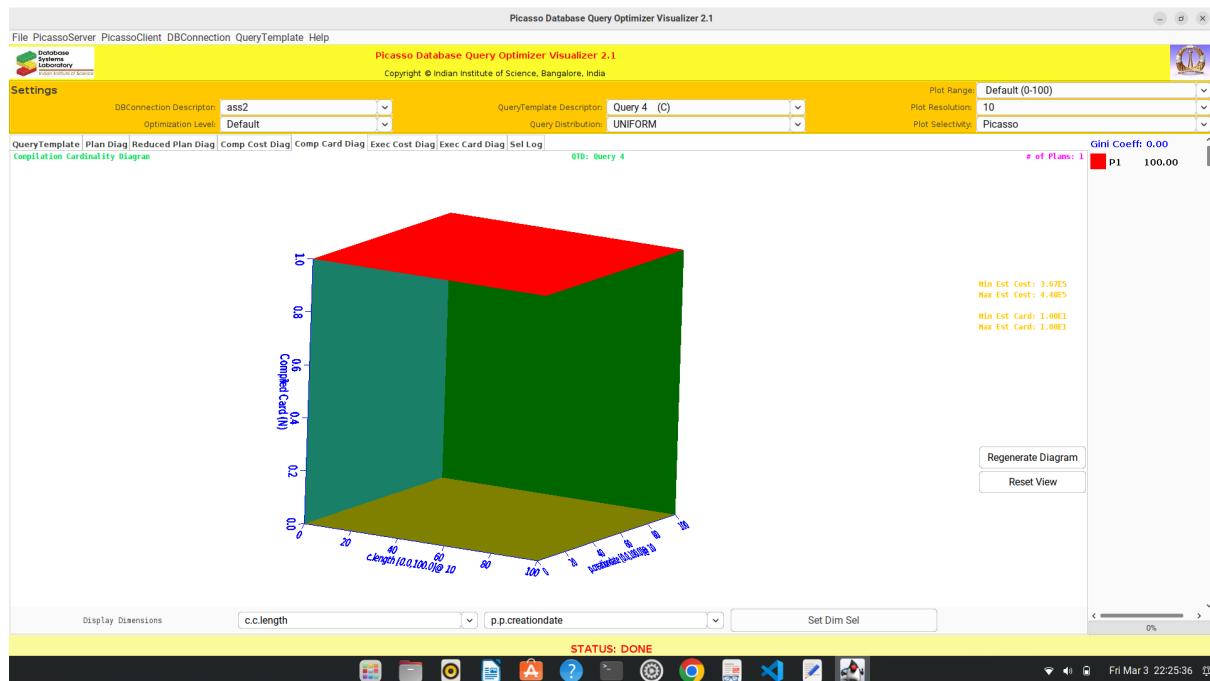
4.3 Compilation Cost diagram

This graph represents the computation time along varying range of length and creationdate along the optimal plan. The value along the z axis in the diagram represents the computation time. The maximum computation time is when we use the whole range for both length and Creationdate. It shows dependence on both length and creationdate.



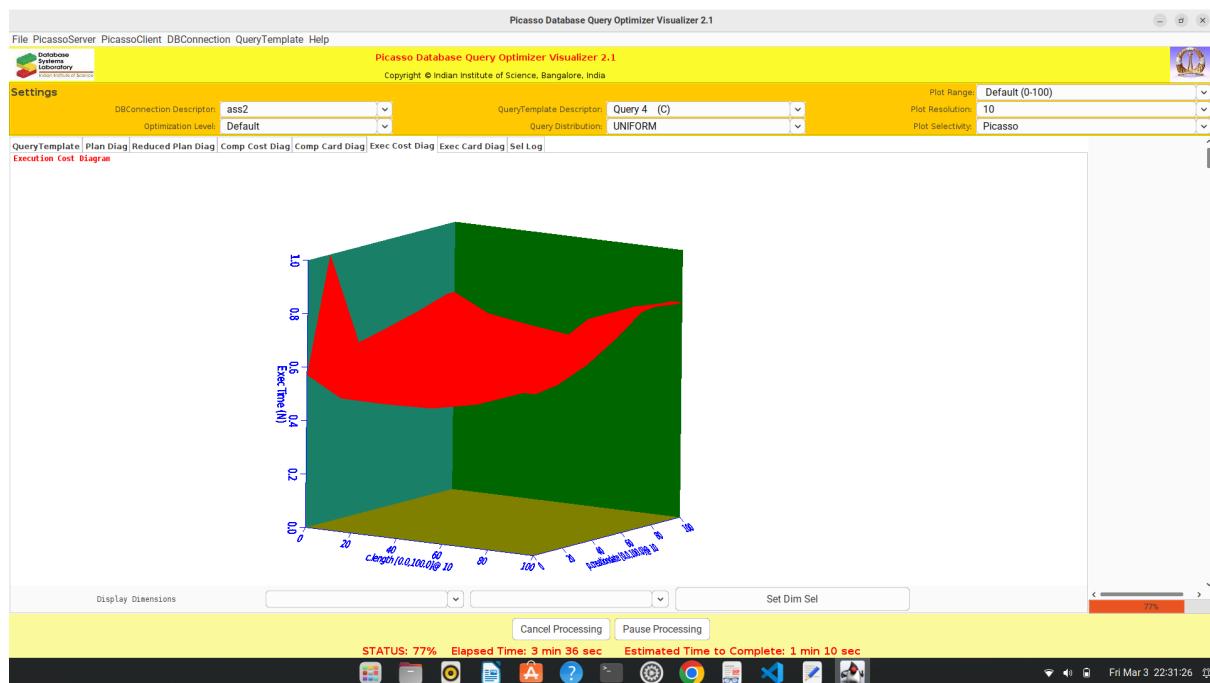
4.4 Compilation Cardinality Diagram

This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1. From the above diagram we can see that the number of distinct values in the query result More or less remains the same with change in predicate values.



4.5 Execution Cost diagram

This diagram represents the runtime query response times for different portions of the tables used. From the diagram it appears that when we use small portion of both the tables the Response time is quite high. When we use almost 50% values of both the tables the Response time for the query is minimum.. Also when we use large portion of both tables We get high response time.



4.6 Plan Diagram

Plan

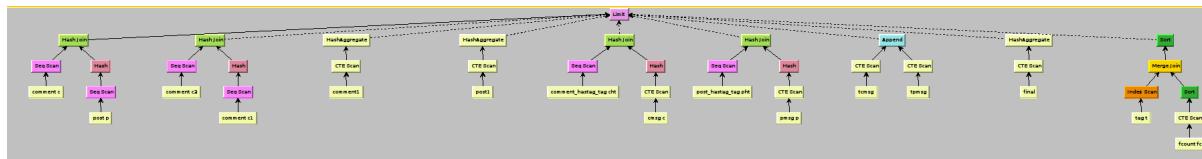
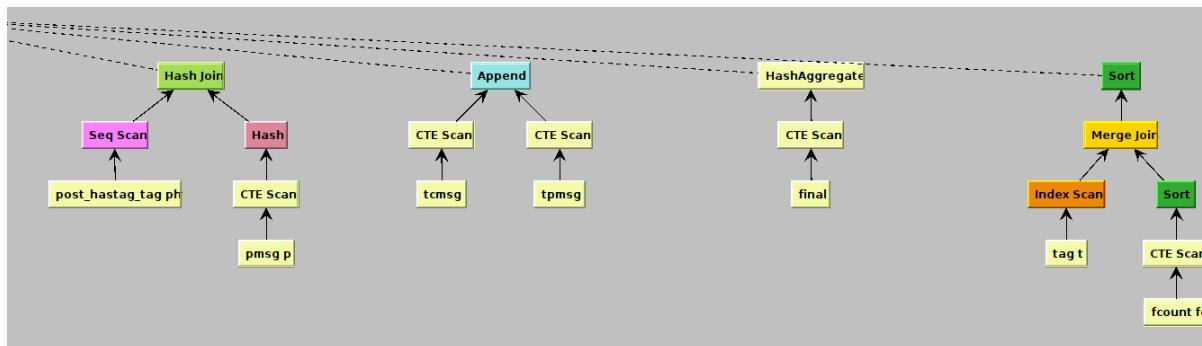
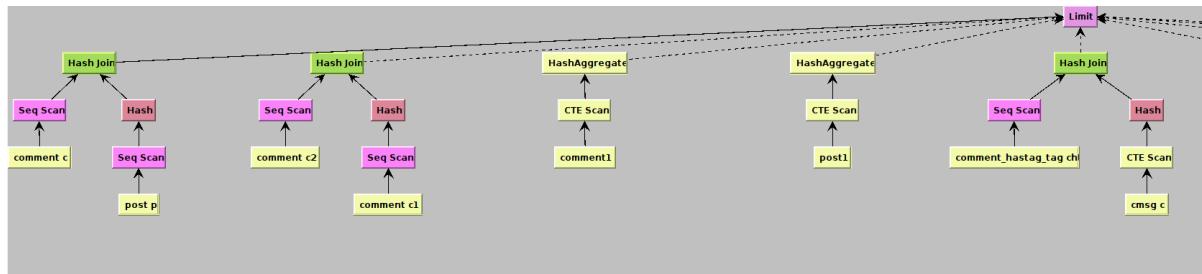


Fig: Total plan diagram



The plan diagram is quite balanced as the height is relatively same across the different Subtrees. The common steps used are hash joins, CTE scans and sequential scan of tables Along with index scan in one subtree which uses sorting so here merge join is a better option.

5. QUERY 5

5.1 Query template

Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

Settings

DBConnection Descriptor: ass2 QueryTemplate Descriptor: query5 (C) Plot Range: Default (0-100)
Optimization Level: Default Query Distribution: UNIFORM Plot Resolution: 10
Plot Selectivity: Picasso

QueryTemplate| Plan Diag| Reduced Plan Diag| Comp Cost Diag| Comp Card Diag| Exec Cost Diag| Exec Card Diag| Sel Log

QueryTemplate Descriptor: query5 Load QueryTemplate Clear QueryTemplate

```

with country_forum(fd)
(
  select distinct f.id from person p1,place p3,place p2,forum f
  where p1.LocationCityId=p2.id and p3.id=p2.PartOfPlaceId and f.ModeratorPersonId=p1.id and p3.name= 'India' and f.creationdate.varies and p1.creationdate.varies
),
post_Forum(fd)
(
  select distinct cf.id from country_forum cf
  where exists
  (
    select * from post p,post_hashtag_tag pht,tag t,tagclass tc
    where p.ContainerForumId=cf.id and pht.PostId=p.id and pht.TagId=t.id and t.TypeTagClassId=tc.id and tc.name= 'TennisPlayer'
  )
),
tag_count(fd,lcnt) as
(
  select pf.id,ph.tagId,lcnt(*) as cnt
  from post_forum pf,post_hashtag_tag pht,post p
  where p.ContainerForumId=pf.id and pht.PostId=p.id
  group by pf.id,ph.tagId
),
max_tag(fd,cnt) as
(
  select fd,max(cnt)
  from tag_count
  group by fd
),
select tc.id as forumid,f.title as forumtitle,t.name as mostpopulartagname,tc.cnt as count
from tag_count tc,max_tag mttag t,forum f
where tc.id=mttag.id and tc.cnt=mttag.cnt and tc.id=f.id and t.id=tc.id
order by count desc,forumid asc,forumtitle asc,mostpopulartagname asc;

```

STATUS: DONE

5.2 Execution Plan Diagram

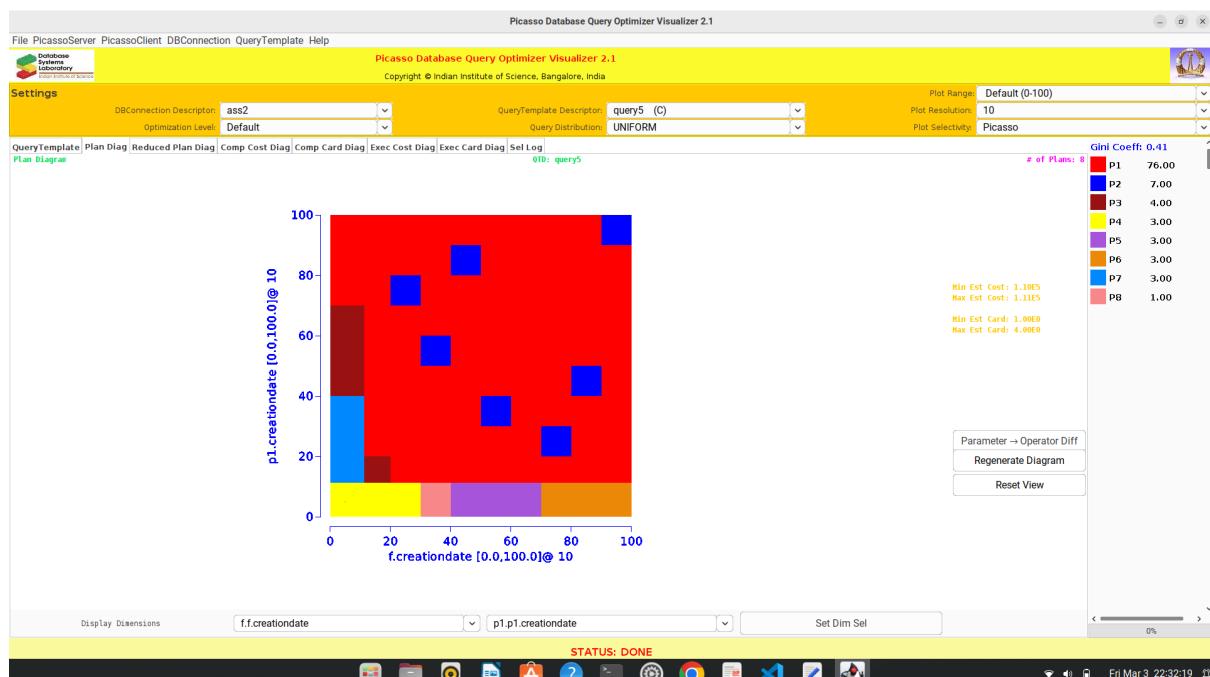
This diagram represents the plan that is optimal at various percentages of the varying tables

For eg here 8 plans are generated for different values of range of length and creationdate

Which means that if we take 50% of person creationdate column and 50% of

forum creationdate column then the

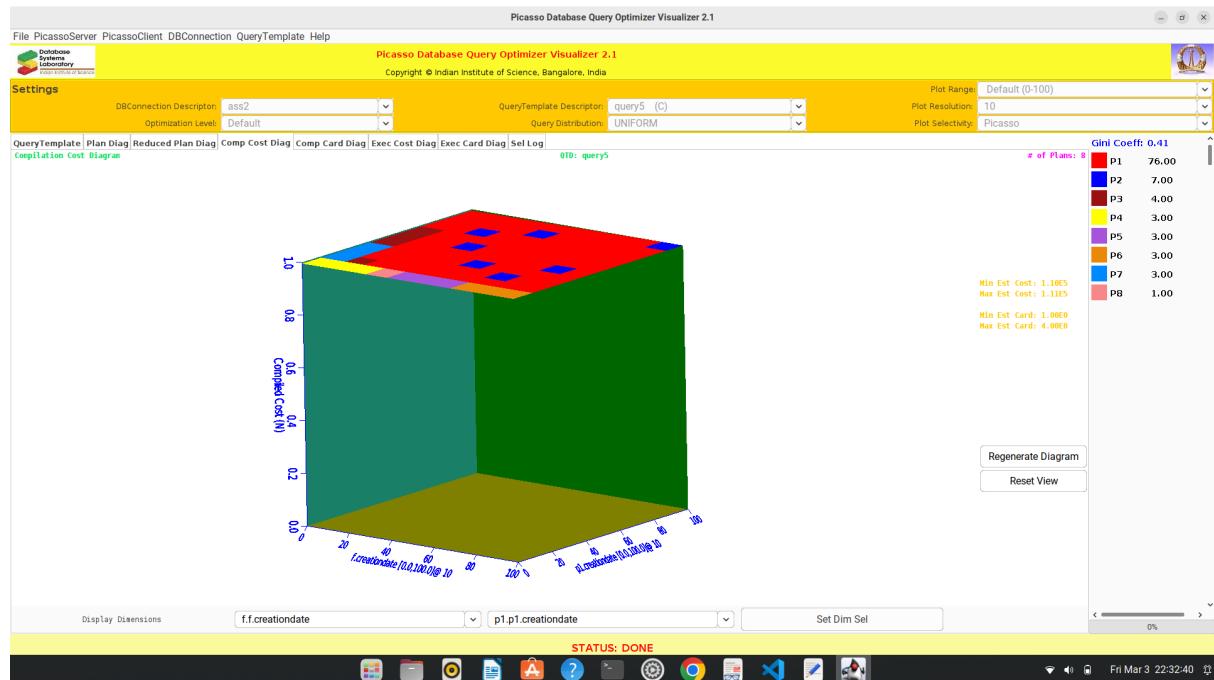
Plan P1 is the most optimal as shown by the red value. Similarly we can also deduce the optimal plans for other regions also. The presence of multiple plans implies that we can still Optimise the query for cases when either of the tables has a lower number of values.



5.3 Compilation Cost diagram

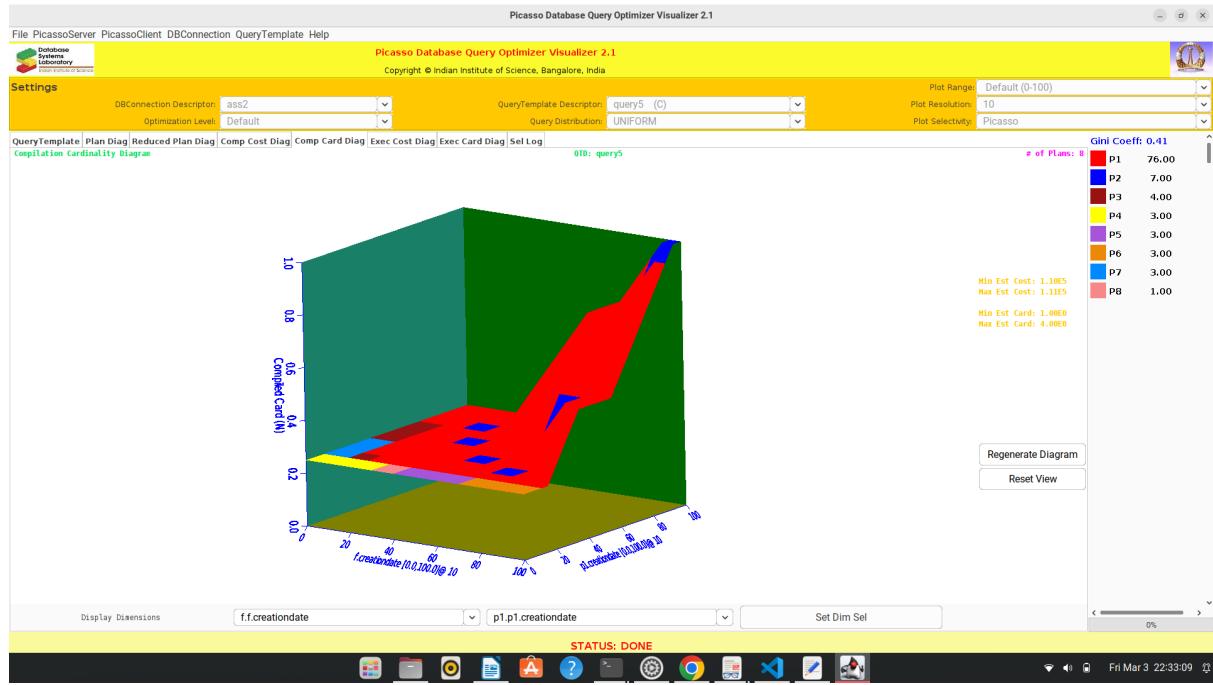
This graph represents the computation time along varying range of post creationdate and comment creationdate along the optimal plan. The value along the z axis in the diagram represents the computation time. The maximum computation time is when we use the whole range for creationdate Values of both the predicates

.The computation time mostly remains same with change in the predicates.



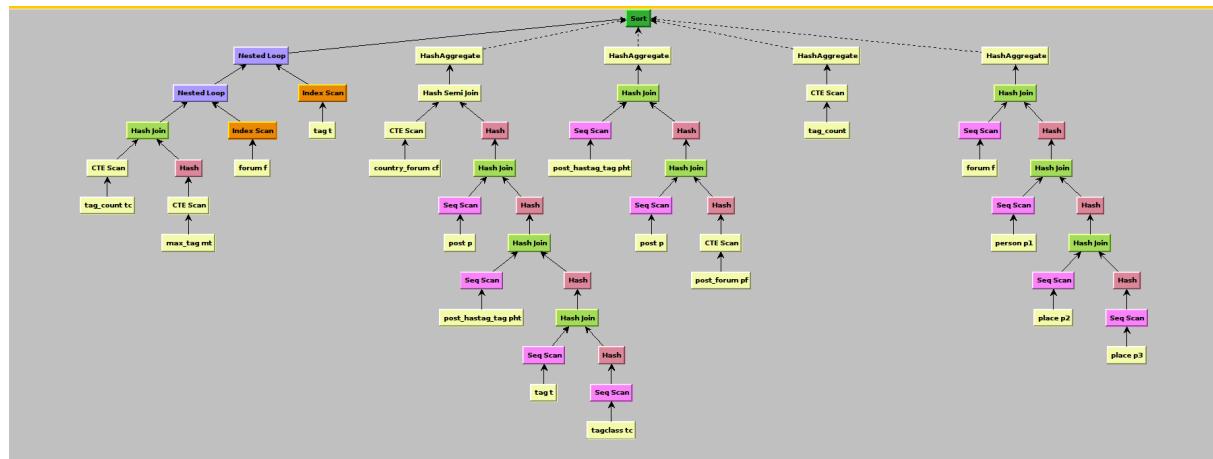
5.4 Compilation Cardinality Diagram

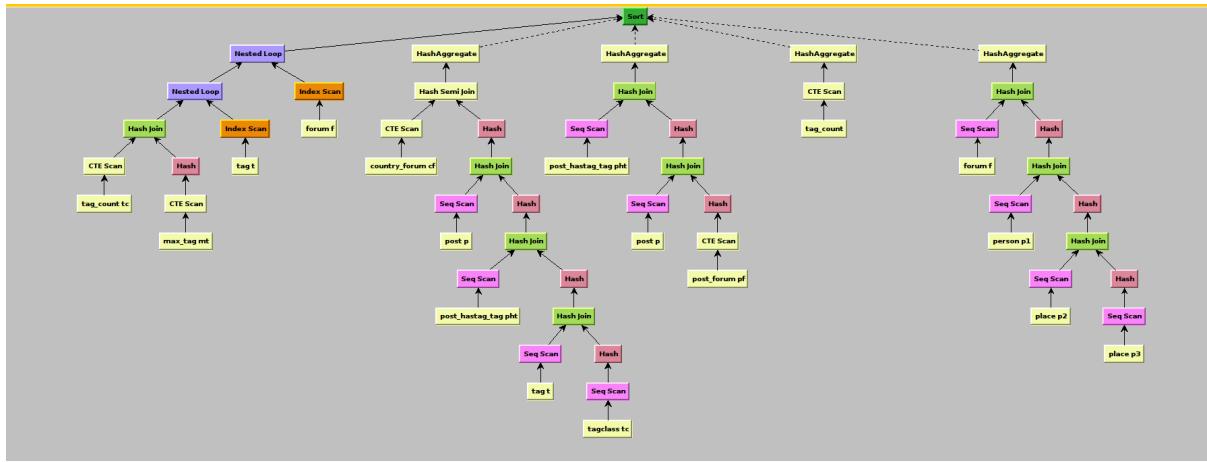
This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1. From the above diagram we can see that the number of distinct values in the query result The number of distinct values in the result increases with increase in portions of the relation Predicate tables used.



5.5 Plan Diagrams

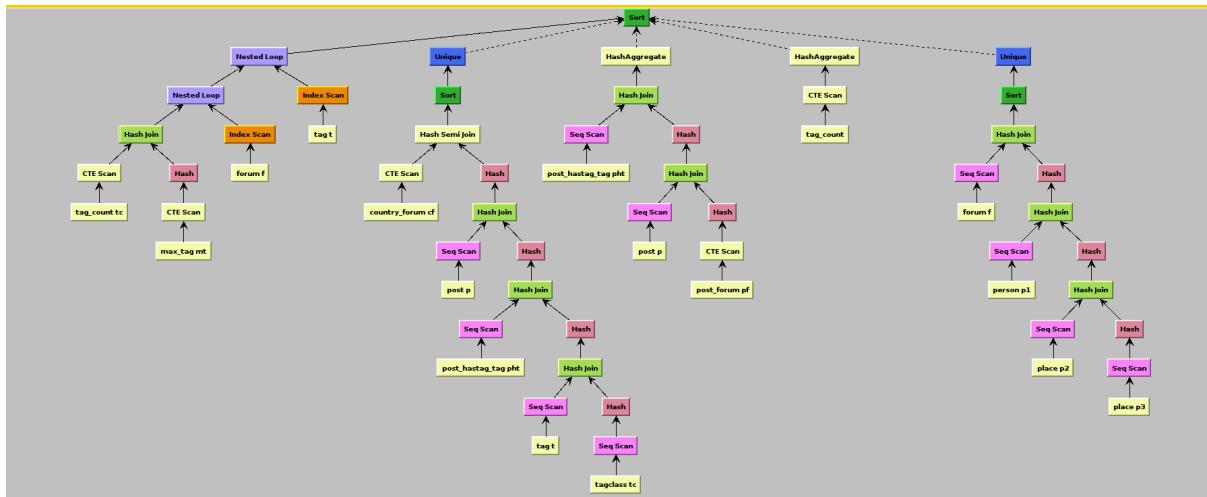
Plan 1





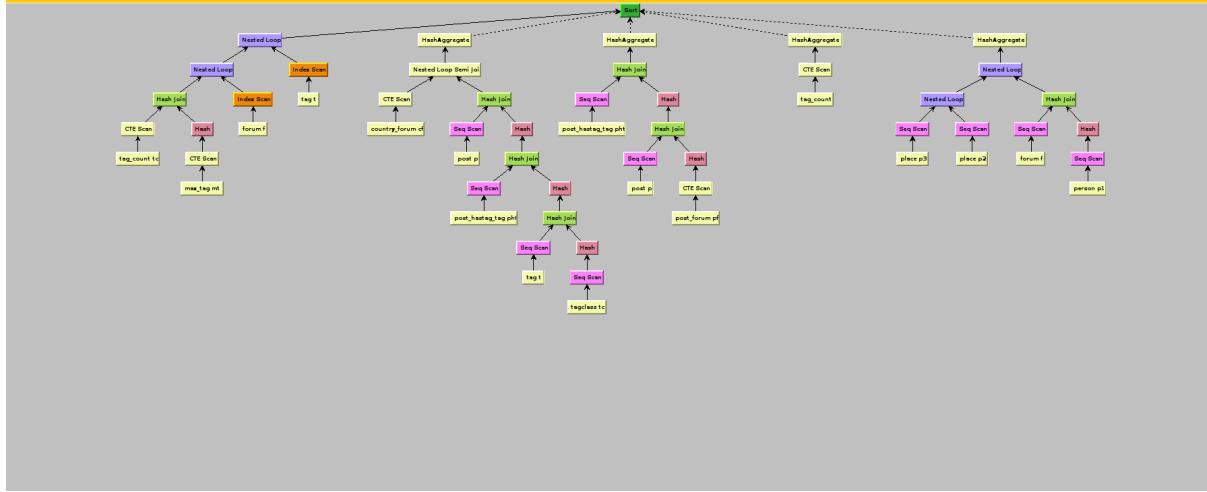
The plan diagram is quite balanced except along two subtrees where it becomes a little skewed as the height is relatively same across the different subtrees except few. The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 3



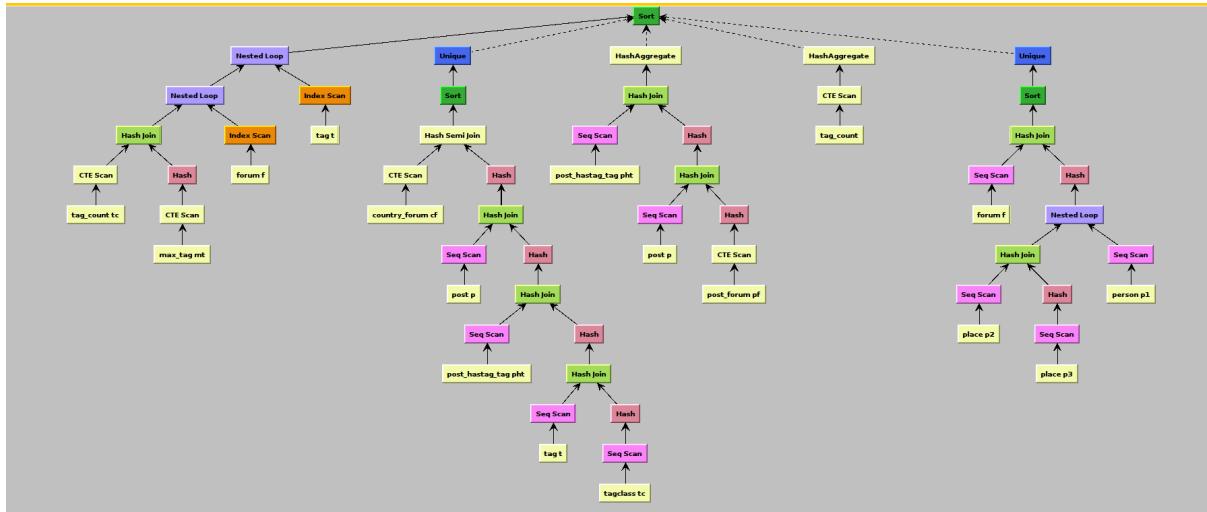
The plan diagram is quite balanced except along two subtrees where it becomes a little skewed as the height is relatively same across the different subtrees except few. The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 4



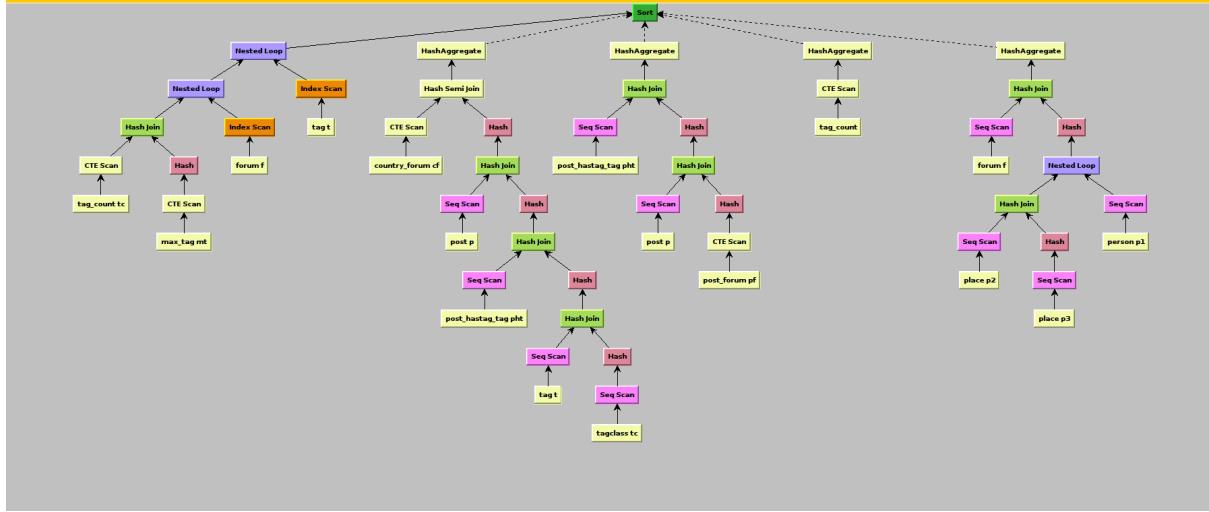
The plan diagram is quite balanced except along one subtree where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .The subtree which was previously unbalanced Is now somewhat balanced.

Plan 5



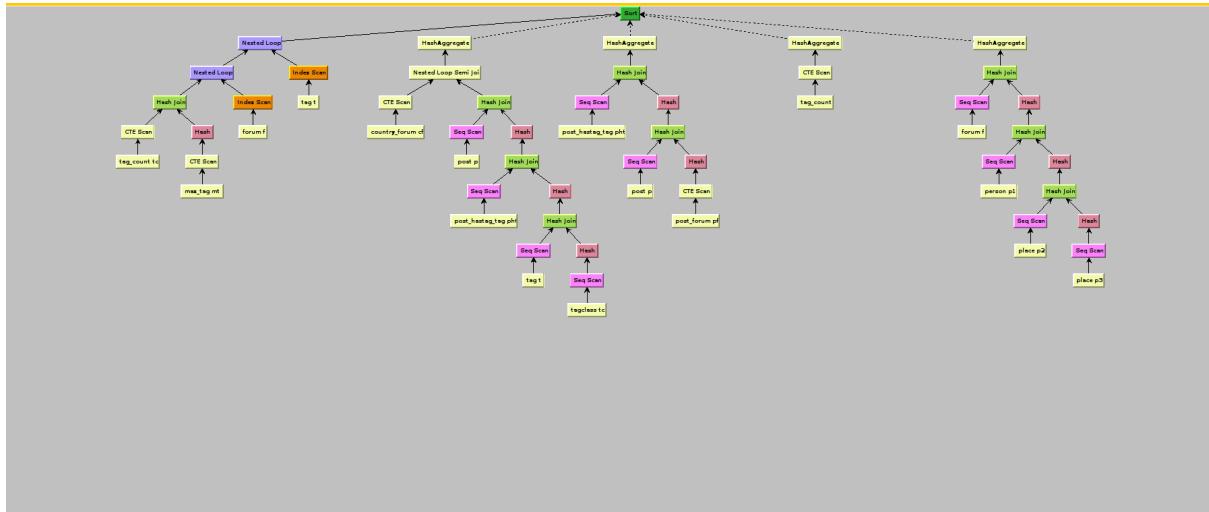
The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 6



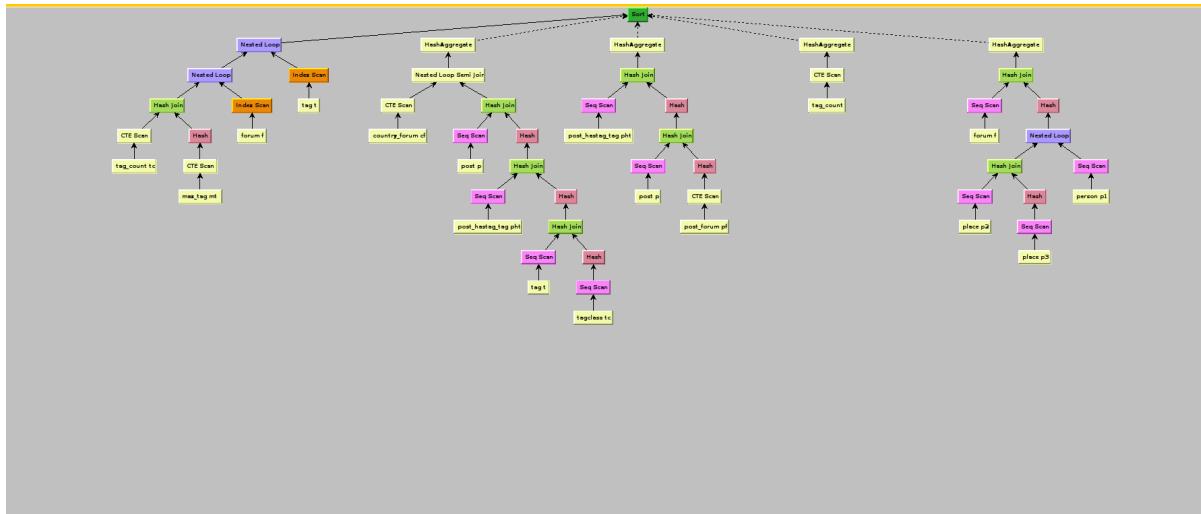
The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 7



The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 8



The plan diagram is quite balanced except along two subtrees where it becomes a little skewed as the height is relatively same across the different subtrees except few. The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

QUERY OPTIMISING

Query 1

Since we have to output in the format person1sid<person2sid hence in the table Where we have find people having at least K tag in common we can limit the table From preventing it to grow by symmetric tuples by imposing a less than relation. It reduces the time to 50%. Indexing doesn't helped much.

Query 2

Here joining person_knows_person directly with person1 and person2 to find friends
Having common birthmonth and same university instead of finding them separately
Reduced time from 180 ms to 20 ms. Here I also tried indexing but it didn't help much.

Query 3

Here I have used indexing on 3 attributes from post_hashtag_tag, comment_hashtag_tag and tag tables as these attributes are quite extensively used in comparisons and joins which take up a lot of time if not indexed . By using indexing we could reduce the time by 50%.

Query 4

Here I have used indexing on tables post,comment ,comment_hashtag_tag and post_hashtag_tag and tag tables as these tables have been widely used in joins and also the tables are of large size which may affect efficiency . I have also tried to improve efficiency by removing unnecessary distincts and select * as they take up large time.and union all instead Of union operator which takes large time because of removing duplicates.

Query 5

Here the query written was much optimal in beginning itself and using index on tables like Place person and forum tables gave slight improvement in time. Further improvements can be obtained by using using unnecessary distincts and slight modification in logic.