DBMS
Assignment – 2

Rishita Agrawal (2020CS50439)
Rajat Bhardwaj (2020CS50436)

Query – 1

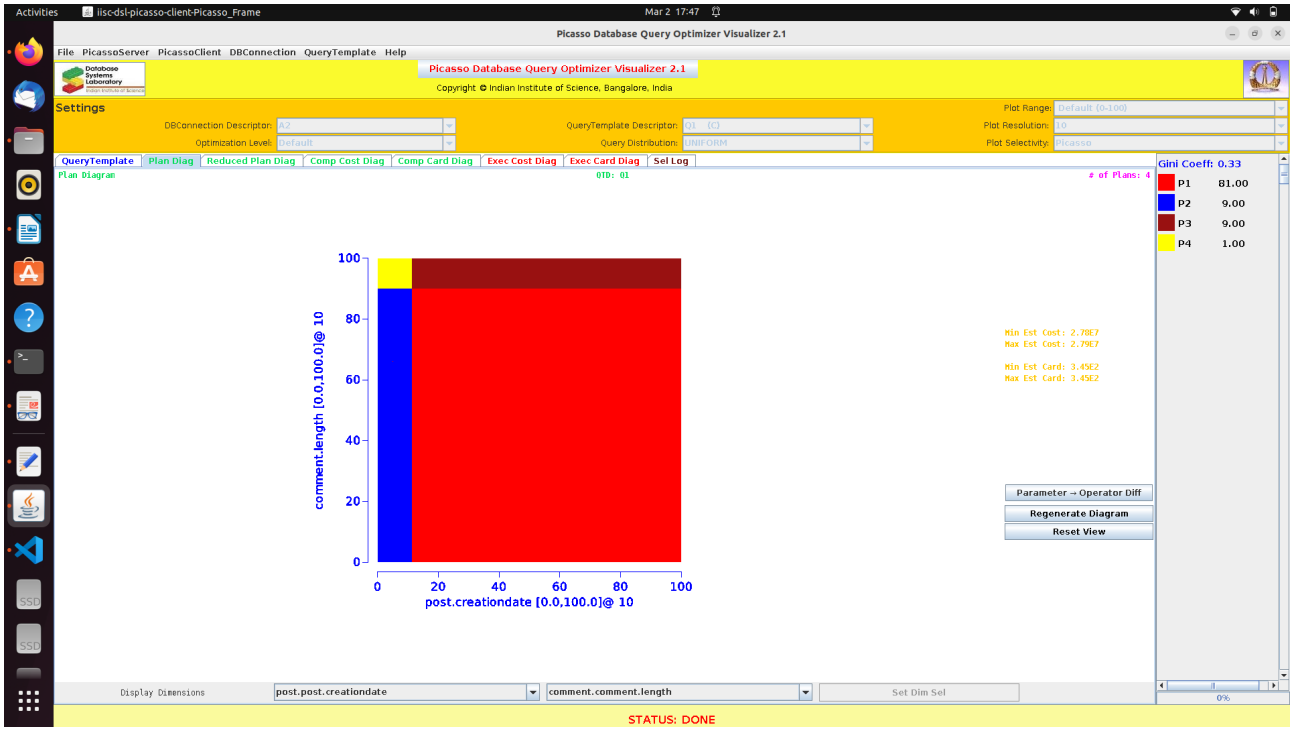
Query Template

```
WITH personpairs AS (
   SELECT t1.id AS person1sid, t2.id AS person2sid
   FROM person AS t1
   JOIN person AS t2 ON t1.id < t2.id
EXCEPT
   SELECT person1id AS person1sid, person2id AS person2sid
   FROM person_knows_person
   WHERE person1id < person2id
EXCEPT
   SELECT person2id AS person1sid, person1id AS person2sid
   FROM person_knows_person
   WHERE person1id > person2id
),
addingtags AS (
SELECT personpairs.person1sid, personpairs.person2sid, t1.tagid
FROM personpairs
JOIN person_hasinterest_tag AS t1 ON t1.personid = personpairs.person1sid
JOIN person_hasinterest_tag AS t2 ON t2.personid = personpairs.person2sid
AND t1.tagid = t2.tagid
JOIN tag ON t1.tagid = tag.id
WHERE ('Frank Sinatra', 'William Shakespeare', 'Elizabeth II', 'Adolf Hitler', 'George W. Bush') ::
text LIKE '%' || tag.name || '%' :: text
),
counttagdupremove AS (
SELECT person1sid, person2sid, SUM(tagcount)
FROM
   (
   SELECT person1sid, person2sid, count(DISTINCT tagid) AS tagcount
   FROM addingtags
   GROUP BY person1sid, person2sid
   UNION
   SELECT person1sid, person2sid, 0 AS tagcount
   FROM personpairs
   ) AS counttag
GROUP BY person1sid, person2sid
HAVING SUM(tagcount) >= 2
),
addingfriends AS (
   SELECT counttagdupremove.person1sid, counttagdupremove.person2sid, t1.person2id AS friend
   FROM counttagdupremove
   JOIN person_knows_person AS t1 ON t1.person1id = counttagdupremove.person1sid
   JOIN person_knows_person AS t2 ON t2.person1id = counttagdupremove.person2sid
```

```
    AND t1.person2id = t2.person2id
UNION
    SELECT counttagdupremove.person1sid, counttagdupremove.person2sid, t1.person2id AS friend
    FROM counttagdupremove
    JOIN person_knows_person AS t1 ON t1.person1id = counttagdupremove.person1sid
    JOIN person_knows_person AS t2 ON t2.person2id = counttagdupremove.person2sid
    AND t1.person2id = t2.person1id
UNION
    SELECT counttagdupremove.person1sid, counttagdupremove.person2sid, t1.person1id AS friend
    FROM counttagdupremove
    JOIN person_knows_person AS t1 ON t1.person2id = counttagdupremove.person1sid
    JOIN person_knows_person AS t2 ON t2.person1id = counttagdupremove.person2sid
    AND t1.person1id = t2.person2id
UNION
    SELECT counttagdupremove.person1sid, counttagdupremove.person2sid, t1.person1id AS friend
    FROM counttagdupremove
    JOIN person_knows_person AS t1 ON t1.person2id = counttagdupremove.person1sid
    JOIN person_knows_person AS t2 ON t2.person2id = counttagdupremove.person2sid
    AND t1.person1id = t2.person1id
),
countingmutualfriends AS (
SELECT person1sid, person2sid, COUNT(friend) AS mutualfriendcount
FROM addingfriends
GROUP BY person1sid, person2sid
),
addinglikedmessages AS (
    SELECT addingfriends.person1sid, addingfriends.person2sid, post.id as messageid
    FROM addingfriends
    JOIN post ON post.creatorpersonid = addingfriends.friend
    JOIN person_likes_post AS t1 ON t1.personid = addingfriends.person1sid
    AND t1.postid = post.id
    JOIN person_likes_post AS t2 ON t2.personid = addingfriends.person2sid
    AND t2.postid = post.id
    WHERE post.creationdate < '2011-07-19'
    AND post.creationdate :varies
UNION
    SELECT addingfriends.person1sid, addingfriends.person2sid, comment.id as messageid
    FROM addingfriends
    JOIN comment ON comment.creatorpersonid = addingfriends.friend
    JOIN person_likes_comment AS t1 ON t1.personid = addingfriends.person1sid
    AND t1.commentid = comment.id
    JOIN person_likes_comment AS t2 ON t2.personid = addingfriends.person2sid
    AND t2.commentid = comment.id
    WHERE comment.length > 100
    AND comment.length :varies
),
countmessagedupremove AS (
SELECT person1sid, person2sid, SUM(messagecount)
FROM
    (
    SELECT person1sid, person2sid, COUNT(messageid) AS messagecount
    FROM addinglikedmessages
```

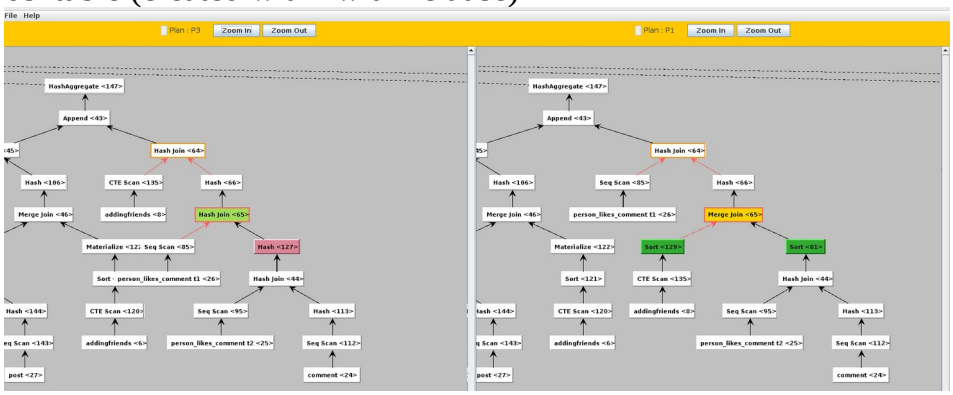
```
    GROUP BY person1sid, person2sid
    UNION
    SELECT person1sid, person2sid, 0 AS messagecount
    FROM counttagdupremove
    ) AS countlikedmessages
GROUP BY person1sid, person2sid
HAVING SUM(messagecount) >= 10
)
SELECT countmessagedupremove.person1sid, countmessagedupremove.person2sid,
countingmutualfriends.mutualfriendcount
FROM countmessagedupremove
JOIN countingmutualfriends ON countmessagedupremove.person1sid =
countingmutualfriends.person1sid
AND countmessagedupremove.person2sid = countingmutualfriends.person2sid
ORDER BY person1sid, mutualfriendcount DESC, person2sid
```

Exact Plan Diagram

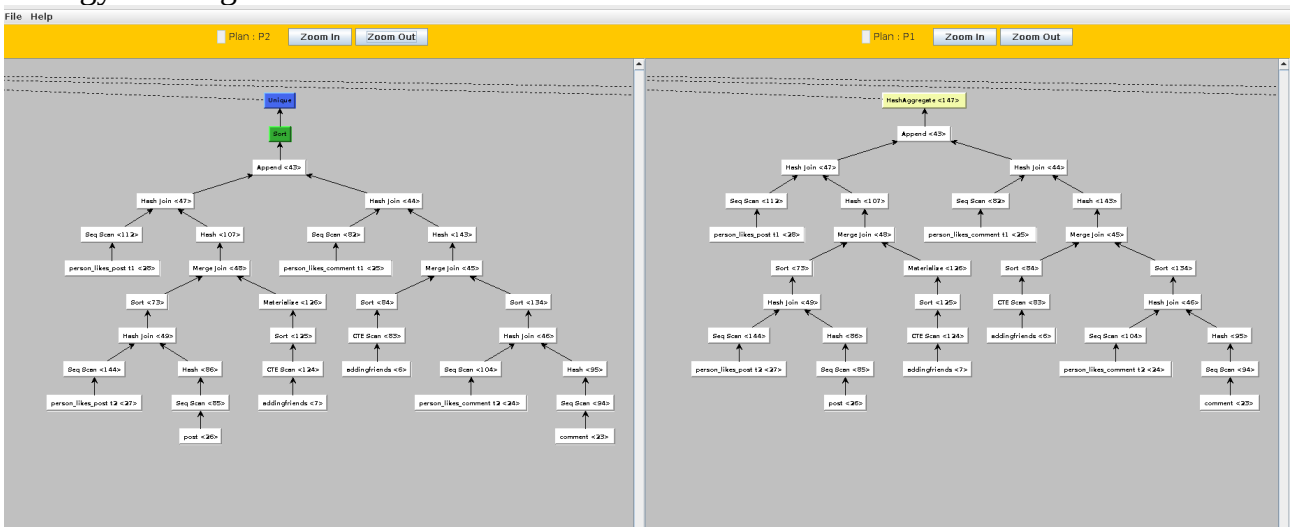


Comments

Since the comment length we have selected is greater than 100, it only starts appearing in the last chunk(100%). Thus it needs to change the strategy to join the comments with the ‘addingfriends’ table (created with “with” clause)



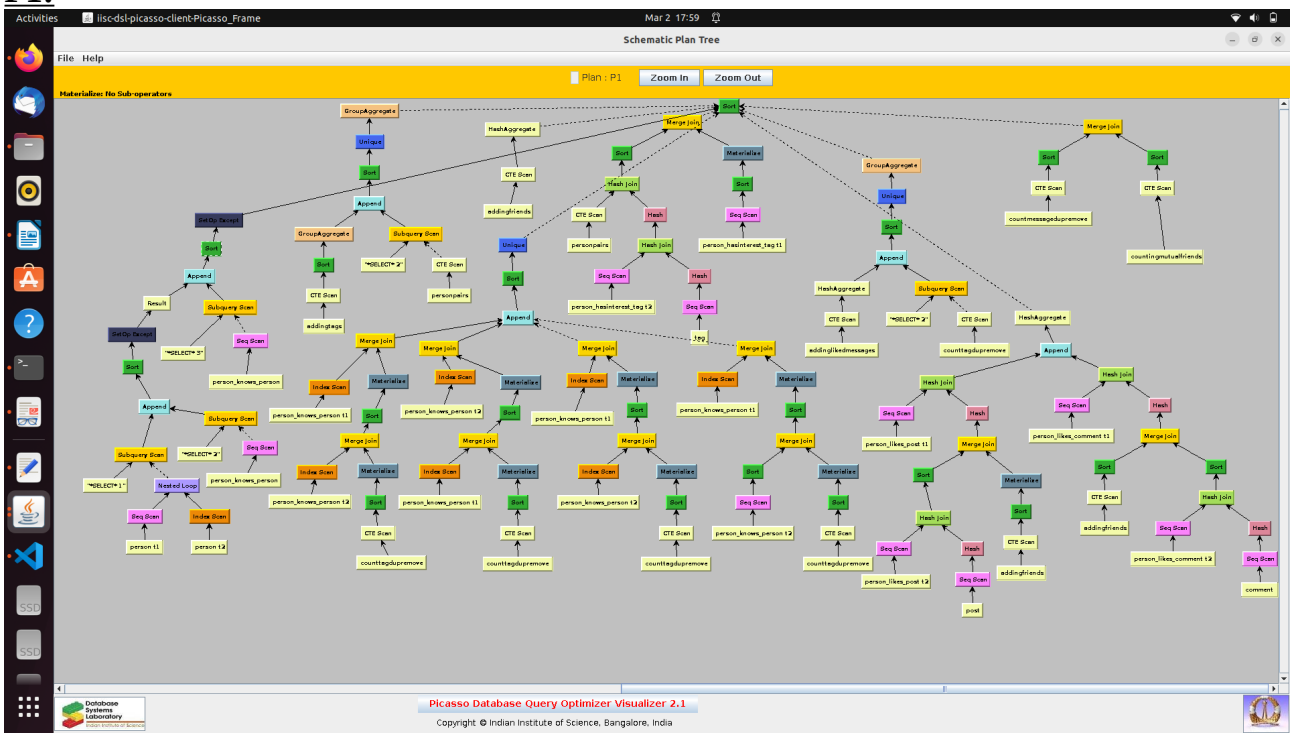
For creationdate vary, the date ‘2011-07-19’ appears in the 20% chunk. Therefore the strategy is changed.



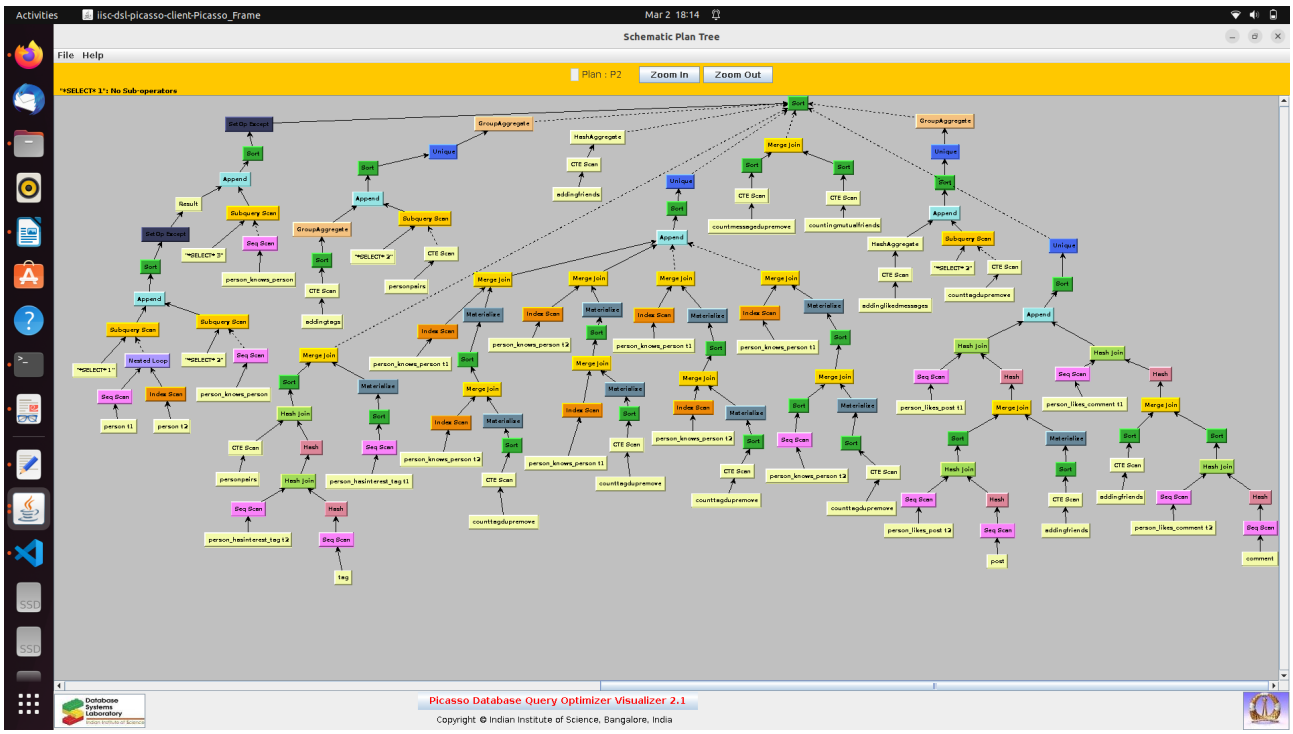
Also in case of 10% chunk of posts and 100% chunk of comments, we obtain comments which fall under the constraints thus there is a different plan tree P4 which changes the joining strategy a bit to optimise the query. Similarly for the 20% chunk of posts and 90% chunk of comments P1 is used.

Plan Trees

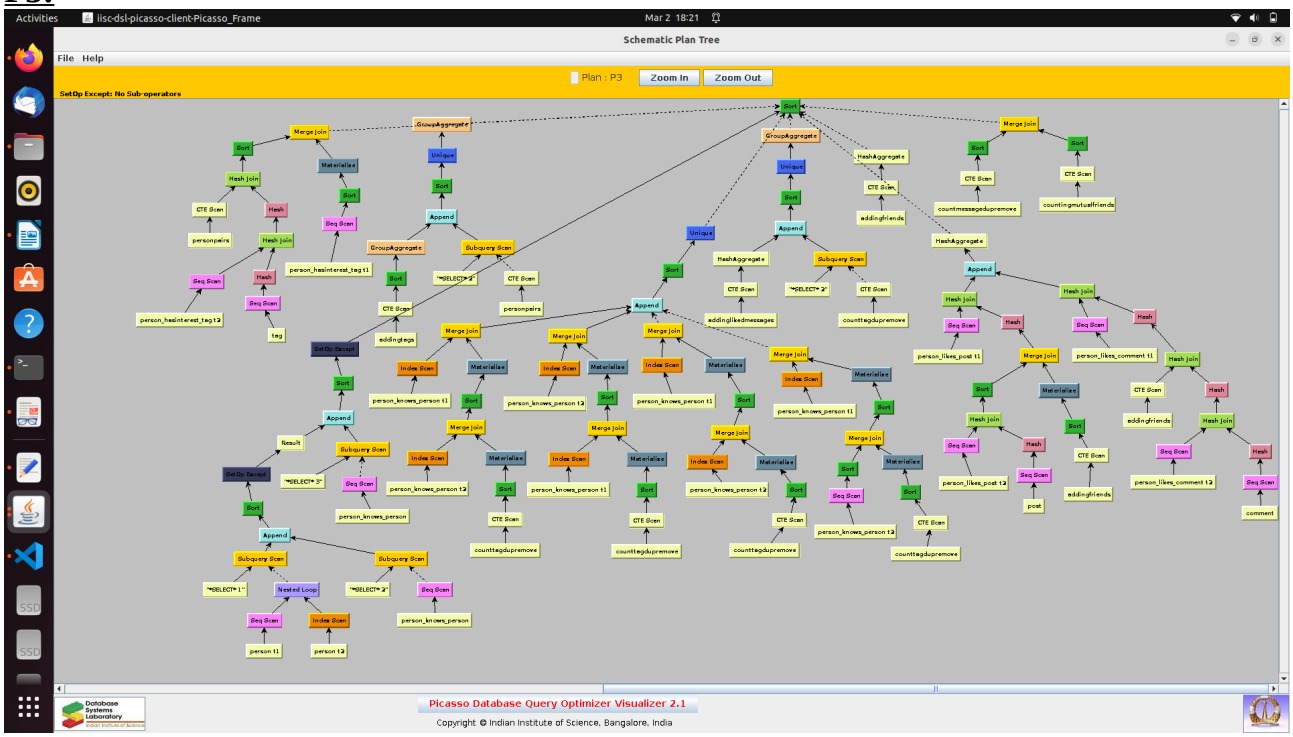
P1:



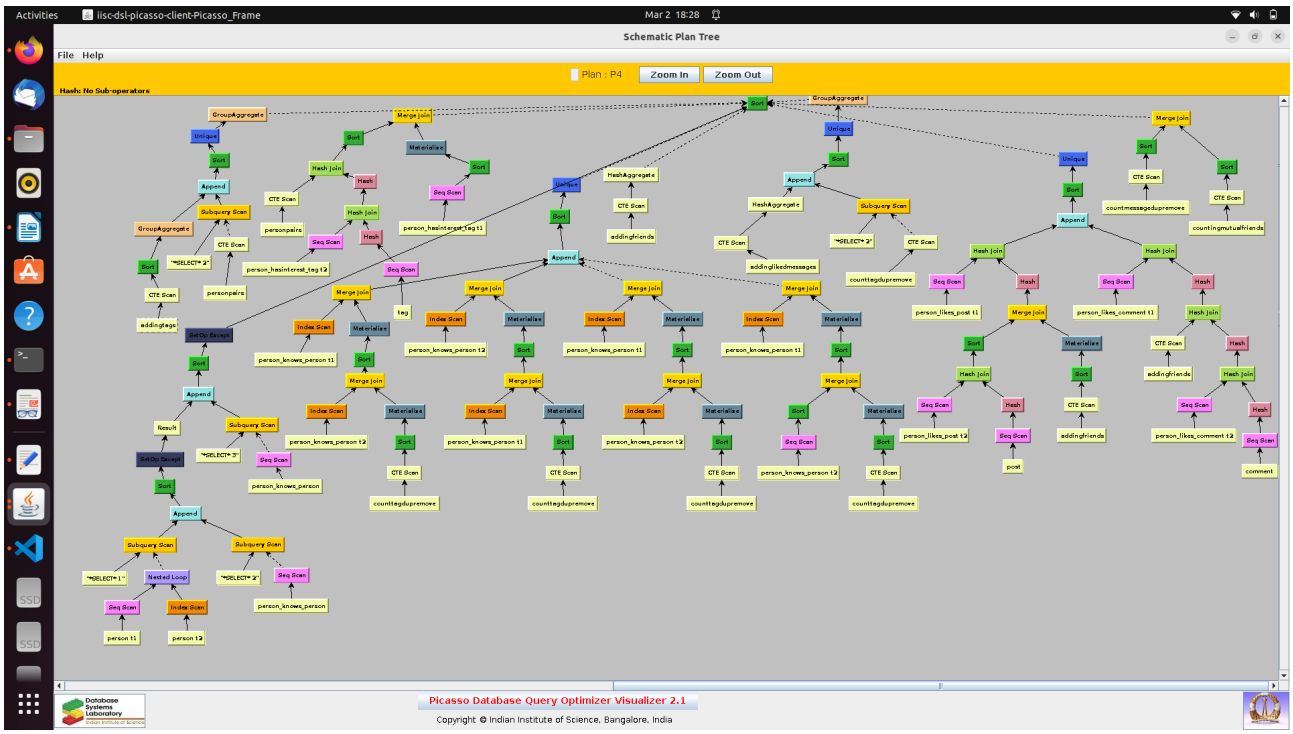
P2:


Activities
iisc-dsl-picasso-client-Picasso_Frame
Mar 2 18:14
Schematic Plan Tree
File Help
Plan : P2
Zoom In
Zoom Out
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

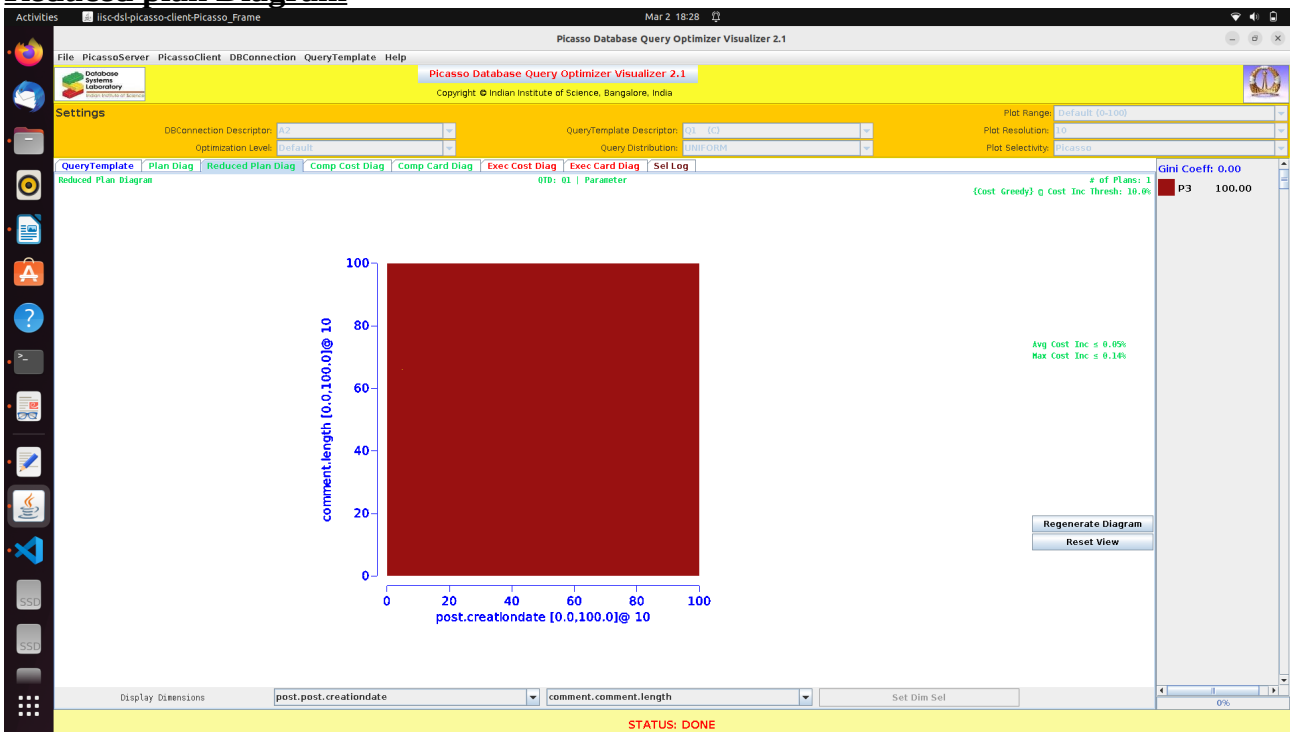
P3:


Activities
iisc-dsl-picasso-client-Picasso_Frame
Mar 2 18:21
Schematic Plan Tree
File Help
Plan : P3
Zoom In
Zoom Out
SetOp Except: No Sub-operators
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

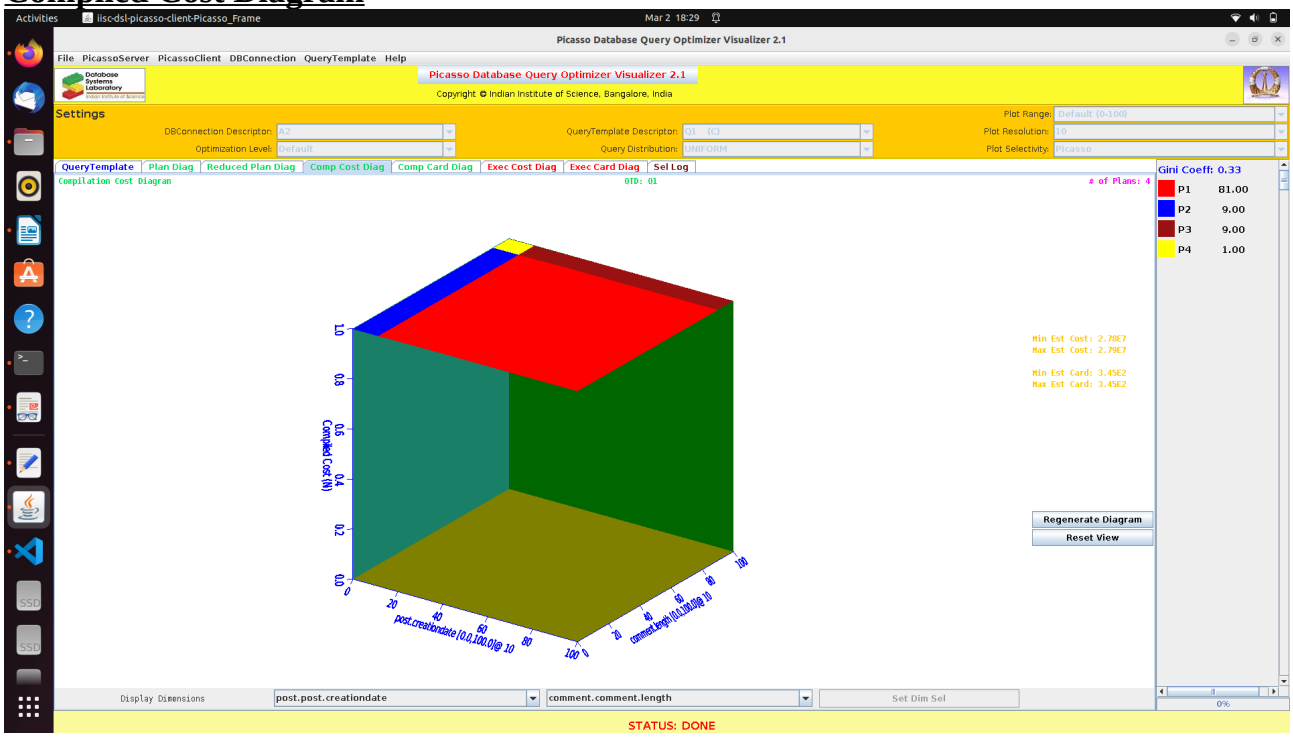
P4:



Reduced plan Diagram



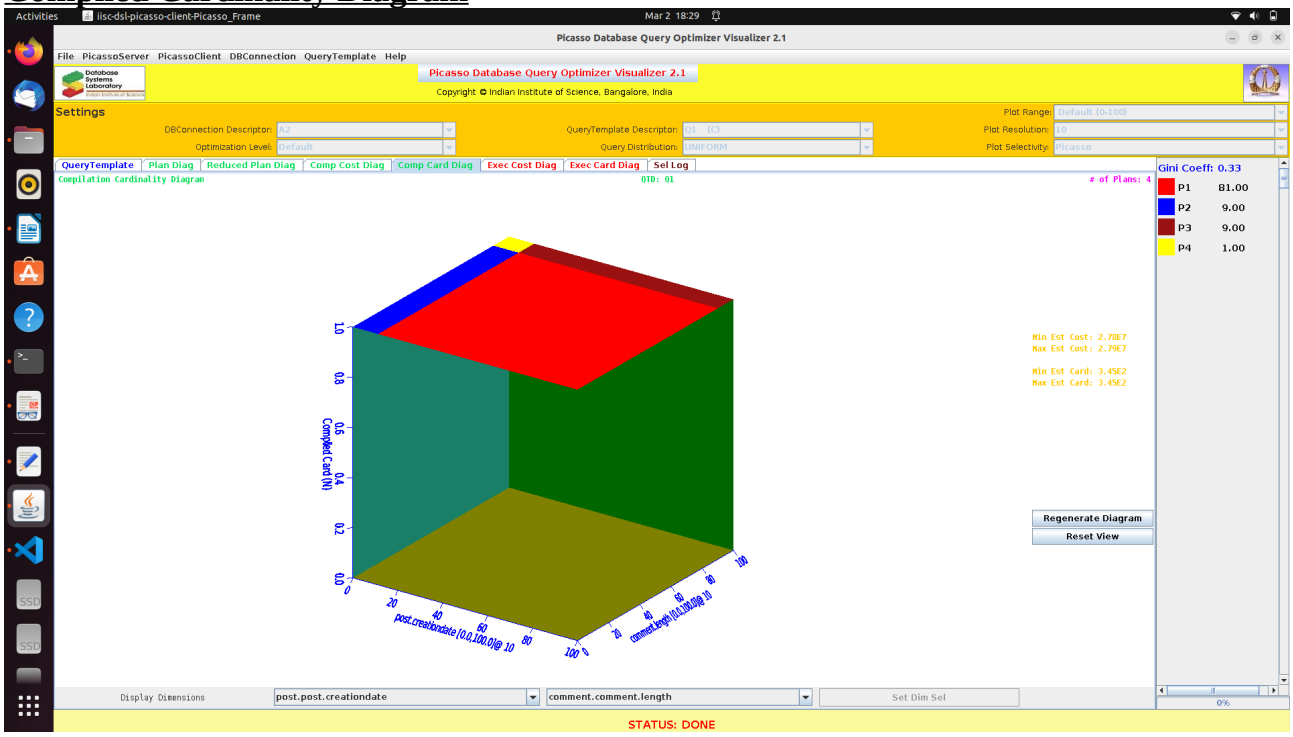
Compiled Cost Diagram



Comments

The cost diagram is (almost) constant for all plans since the first thing we need to do is join person with person (without any constraints) it is the heaviest operation in the entire query(we have optimized this query such that this operation is takes places on filtered people). Thus filtering out the creation date and comment length does not play a role in query runtime.

Compiled Cardinality Diagram



Query – 2

Query Template

```
WITH country AS
(
SELECT id AS country_id
FROM place
WHERE name = 'China'
),
_person as (
select * from person
where person.creationdate :varies
),
accepted_cities AS (
SELECT id AS cityid
FROM place
JOIN country ON country.country_id = place.partofplaceid
),
shortlisted_person AS (
SELECT _person.id AS personid, person_studyat_university.universityid AS universityid,
EXTRACT(MONTH FROM _person.birthday) AS birth_month
FROM _person , person_studyat_university, accepted_cities
WHERE person_studyat_university.personid = _person.id and
_person.locationcityid = accepted_cities.cityid and
_person.creationdate < '2012-07-01'
and _person.creationdate > '2010-06-01'
),
personpairs AS (
SELECT DISTINCT t1.personid AS person1id, t2.personid AS person2id
FROM shortlisted_Person AS t1
JOIN shortlisted_Person AS t2 ON t1.universityid = t2.universityid
AND t1.birth_month = t2.birth_month
AND t1.personid != t2.personid
),
monthanduniversityfilter AS (
SELECT personpairs.person1id, personpairs.person2id
FROM personpairs
JOIN person_knows_person
ON (personpairs.person1id = person_knows_person.person1id AND personpairs.person2id =
person_knows_person.person2id)
OR (personpairs.person2id = person_knows_person.person1id AND personpairs.person1id =
person_knows_person.person2id)
),
p2p3table AS (
SELECT t1.person1id AS person1, t1.person2id AS person2, t2.person2id AS person3
FROM monthanduniversityfilter AS t1
JOIN monthanduniversityfilter AS t2 ON t1.person2id = t2.person1id
and t1.person1id != t2.person2id
UNION
SELECT t1.person1id AS person1, t1.person2id AS person2, t2.person1id AS person3
FROM monthanduniversityfilter AS t1
```

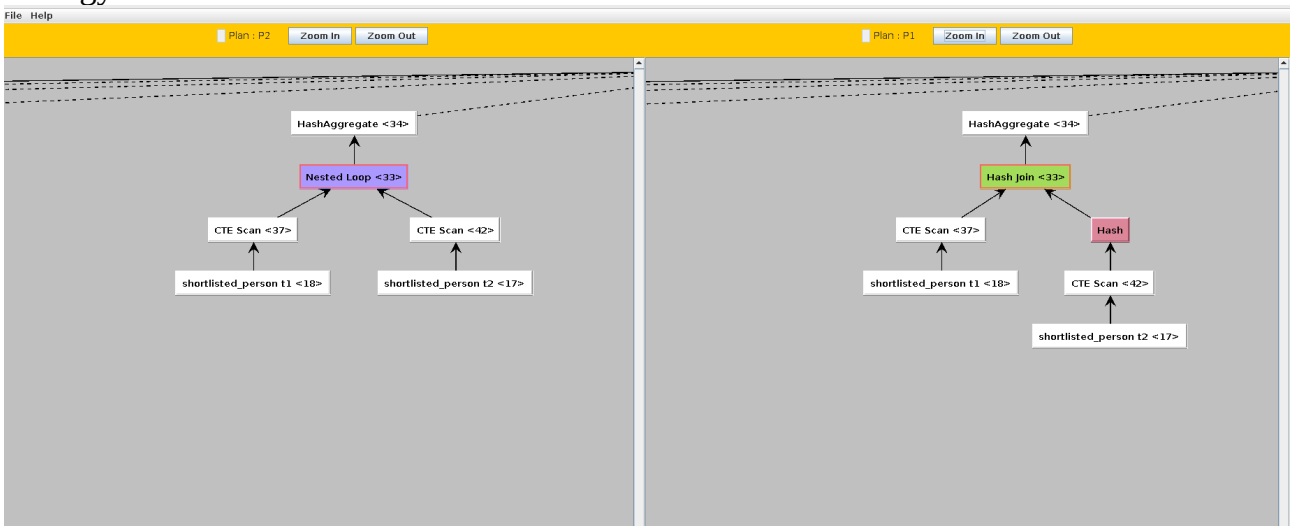
```
JOIN monthanduniversityfilter AS t2 ON t1.person2id = t2.person2id
and t1.person1id != t2.person1id
),
tripletable AS (
SELECT p2p3table.person1, p2p3table.person2, p2p3table.person3
FROM p2p3table
JOIN person_knows_person
ON (p2p3table.person1 = person_knows_person.person1id AND p2p3table.person3 = person2id)
OR (p2p3table.person1 = person_knows_person.person2id AND p2p3table.person3 = person1id)
),
triples AS (
SELECT
CASE
   WHEN person1 <= person2
   and person2 <= person3 THEN ARRAY [person1, person2, person3]
   WHEN person1 <= person3
   and person3 <= person2 THEN ARRAY [person1, person3, person2]
   WHEN person2 <= person1
   and person1 <= person3 THEN ARRAY [person2, person1, person3]
   WHEN person2 <= person3
   and person3 <= person1 THEN ARRAY [person2, person3, person1]
   WHEN person3 <= person2
   and person2 <= person1 THEN ARRAY [person3, person2, person1]
   ELSE ARRAY [person3, person1, person2]
END AS triplets
FROM tripletable
)
SELECT COUNT(DISTINCT triplets)
FROM triples
```

Exact Plan Diagram

Comments

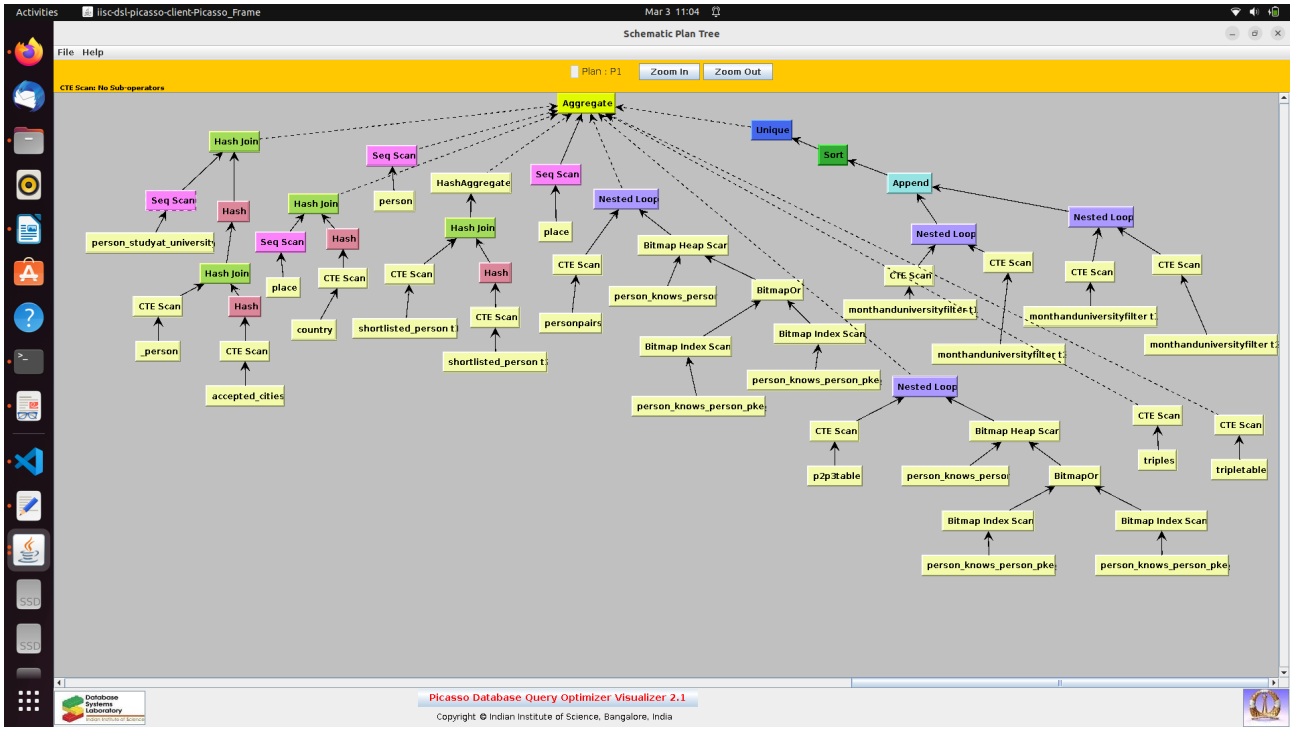
Since in the first 10% chunk of the person table, there are close to 0 number of entries for a person who had a creationdate of more than ‘2010-06-01’, Therefore when we use more rows from the person column, we start getting large number of person accounts with creation date of more than start date thus we need to choose a different combinational strategy

Now after the 50% mark, we get a large number of persons therefore we can change the stratagy more

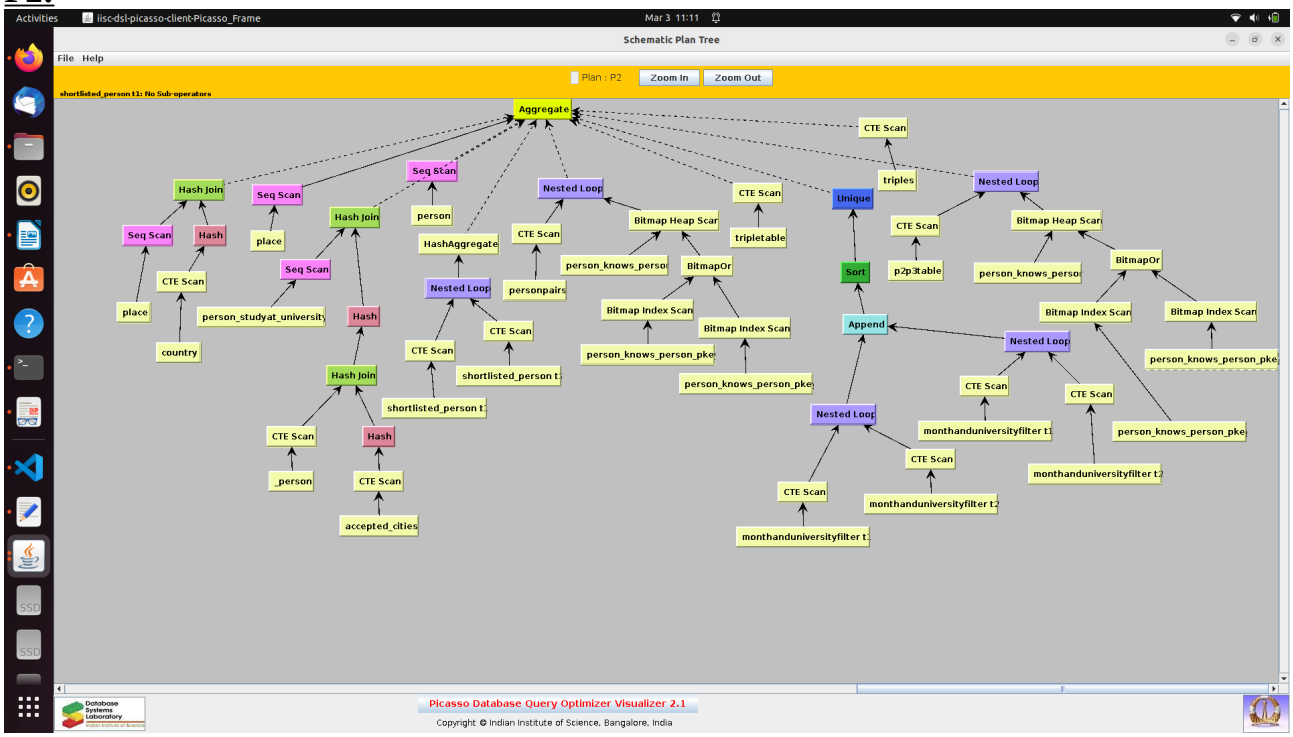


Plan Trees

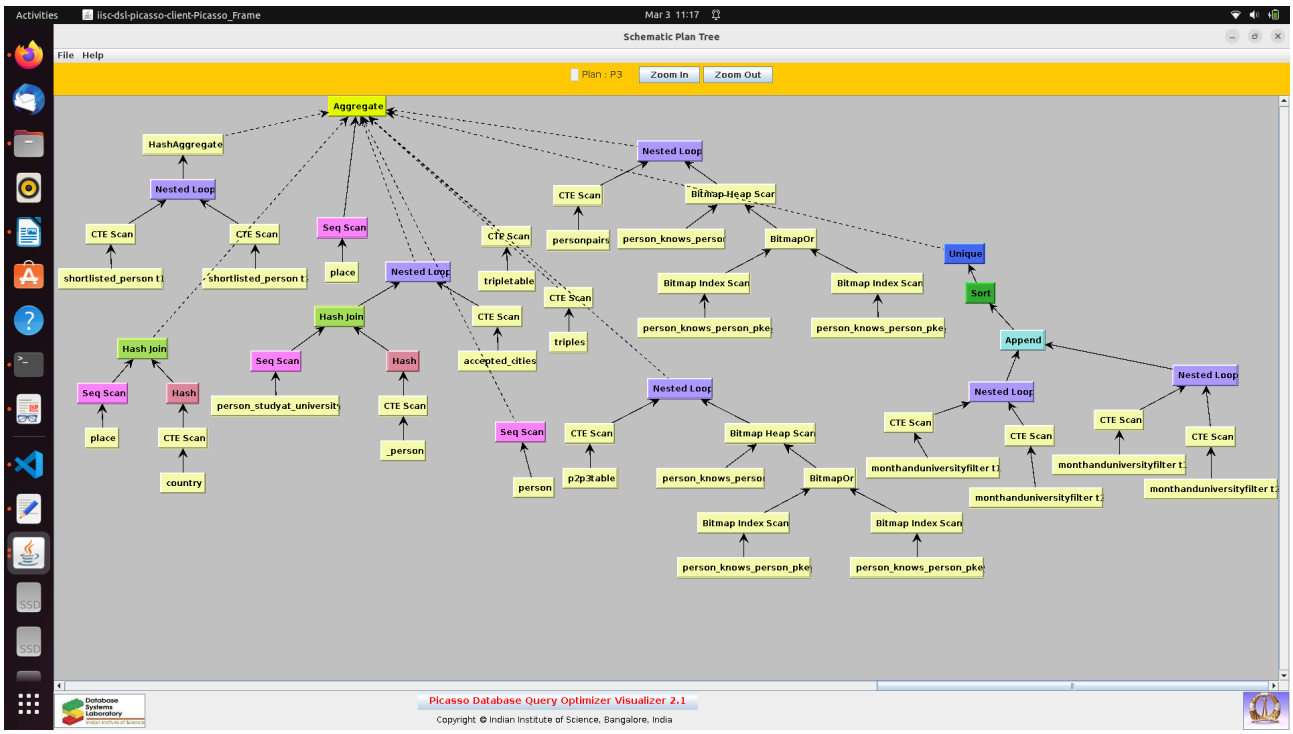
P1:



P2:

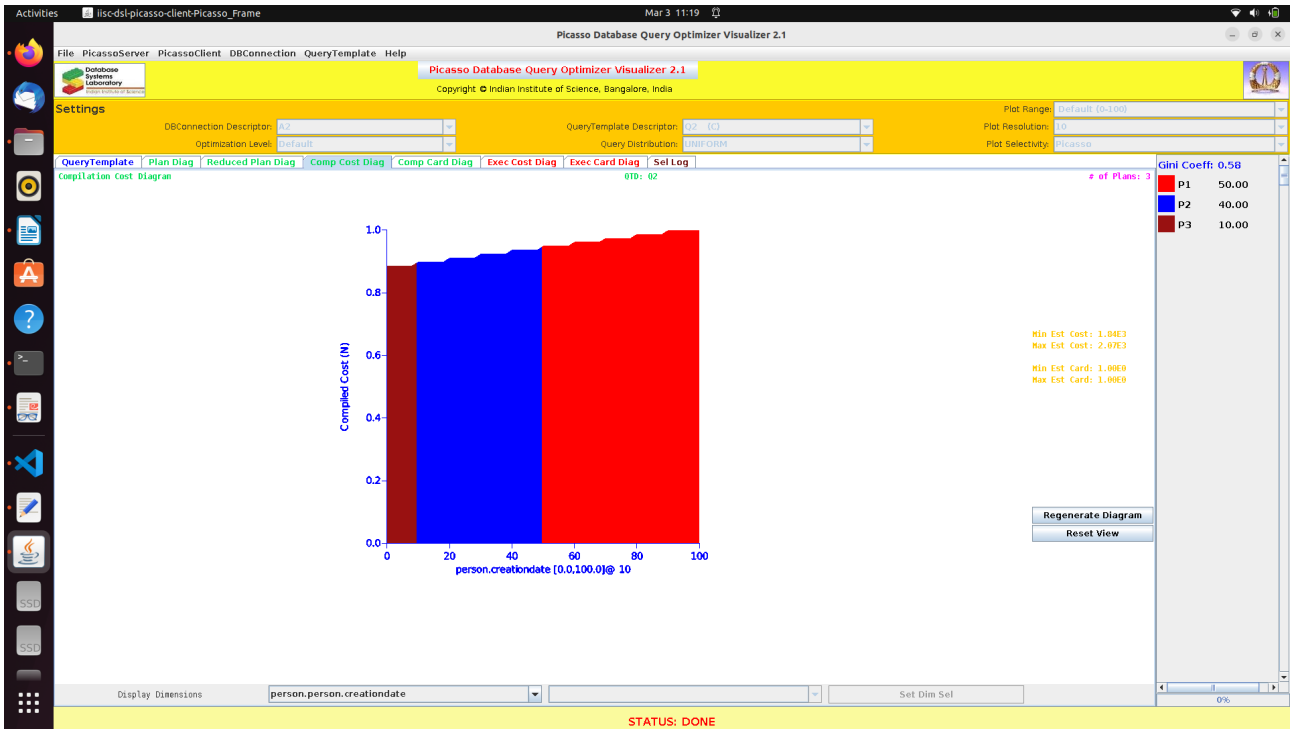


P3:



Reduced Plan Diagram

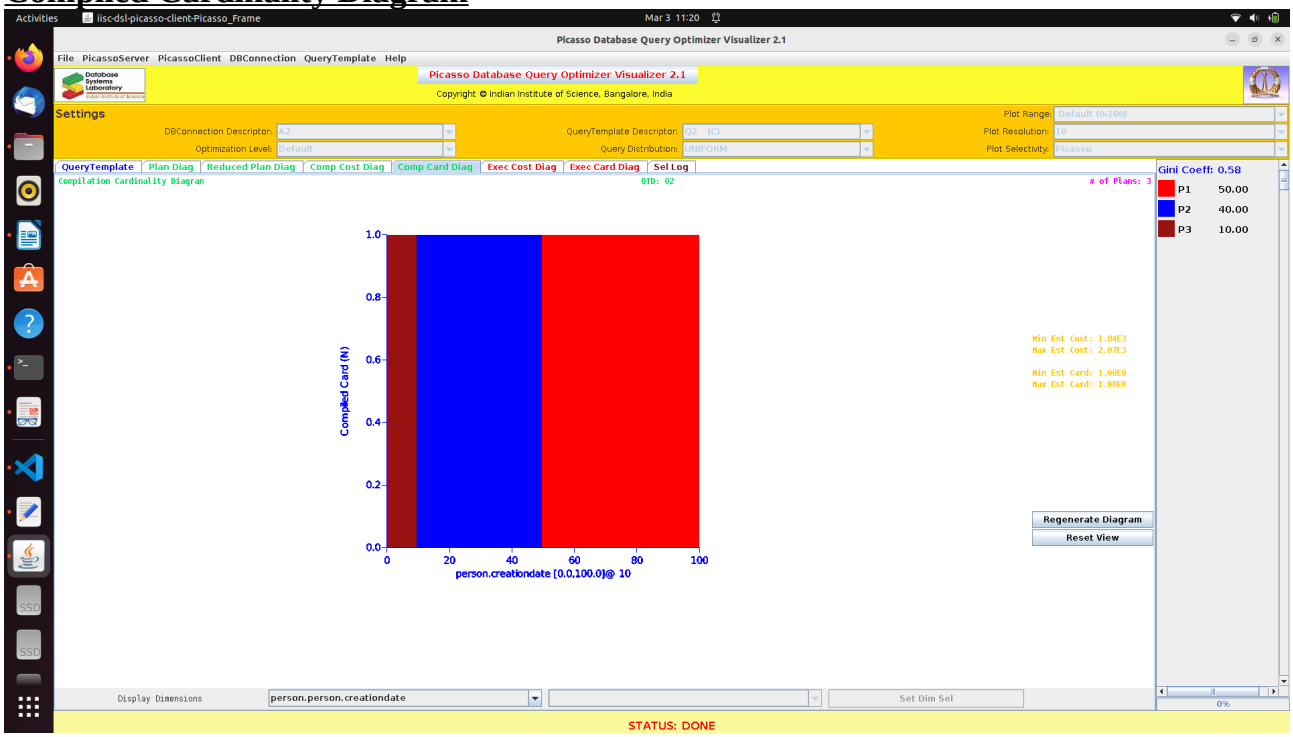
Compiled Cost Diagram



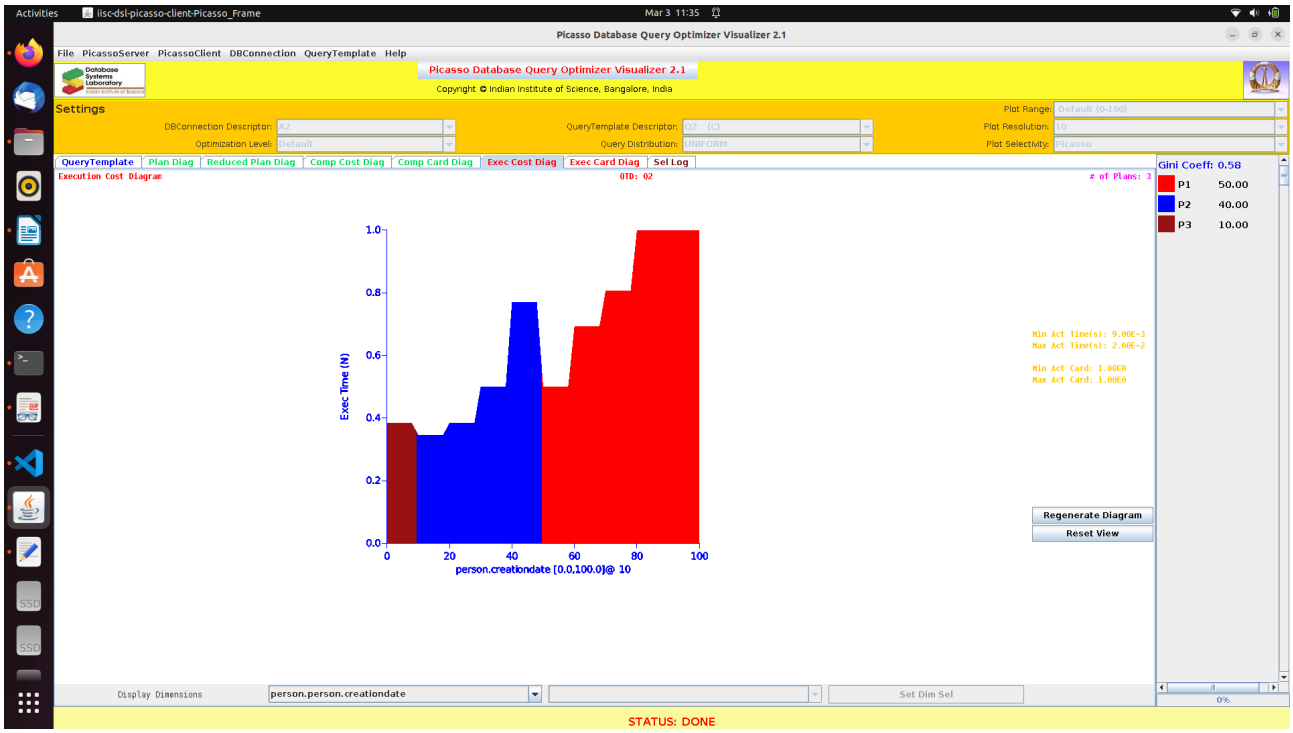
Comments

Cost will increase with increase in the number of filtered people(according to creationdate). Thus as we keep on increasing the chunk (0 – x%) for the creation date we will keep getting more and more people, therefore the overall cost will increase since we need to join these ‘shortlisted_people’ with themselves.

Compiled Cardinality Diagram



Execution Cost Diagram



Query – 3

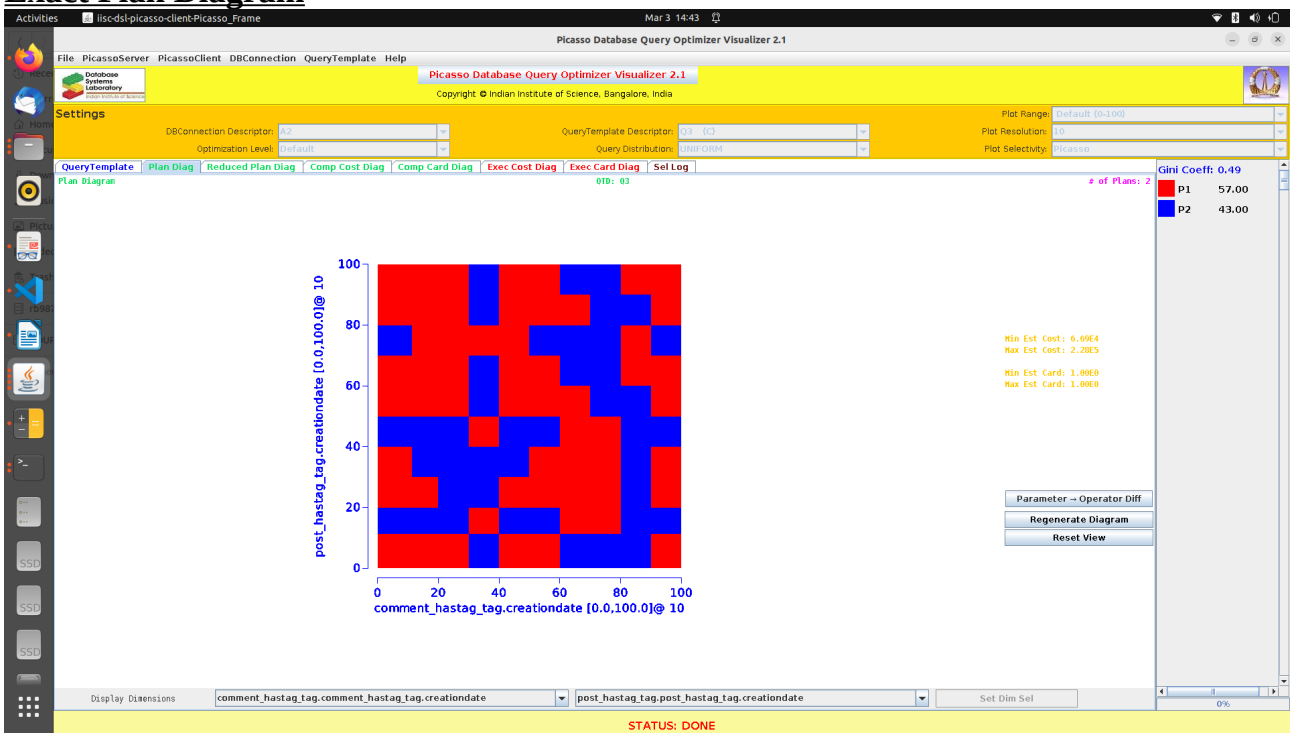
Query Template

```
with
   messages
AS
(
      select
         creationDate, tagid
      from
         comment_hastag_tag
      where

         comment_hastag_tag.creationdate :varies

      union all
      select
         creationDate, tagid
      from
         post_hastag_tag
      where

         post_hastag_tag.creationdate :varies
)
,
   message_create_mid
AS
(
```

```
    SELECT
        tagid, COUNT(tagid) AS ct
    FROM
        messages
    WHERE
        creationDate >= '2010-02-03'
    AND
        creationDate <= '2010-12-03'
    GROUP BY
        tagid

)
,
    message_mid_end
AS
(
    SELECT
        tagid, COUNT(tagid) AS ct
    FROM
        messages
    WHERE
        creationDate >= '2010-12-03'
    AND
        creationDate <= '2011-05-03'
    GROUP BY
        tagid

)
,
    req_tags
AS
(
    SELECT
        t1.tagid
    FROM
        message_create_mid t1 , message_mid_end t2
    WHERE
        t1.tagid = t2.tagid
    AND
        t1.ct >= 5 * t2.ct

)
,
    tag_class_id
AS
(
    SELECT
        typetagclassid, COUNT(typetagclassid) AS ct
    FROM
        tag, req_tags
    WHERE
        tag.id = req_tags.tagid
```

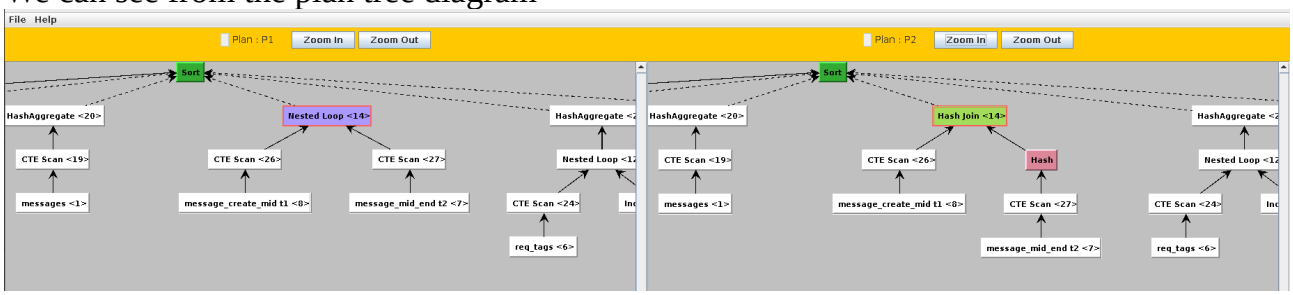
```
    GROUP BY
        typetagclassid

)
SELECT
    name as tagclassname, ct as count
FROM
    tag_class_id, TagClass
WHERE
    tag_class_id.typetagclassid = TagClass.id
ORDER BY
    count DESC, tagclassname;
```

Exact Plan Diagram



Comments

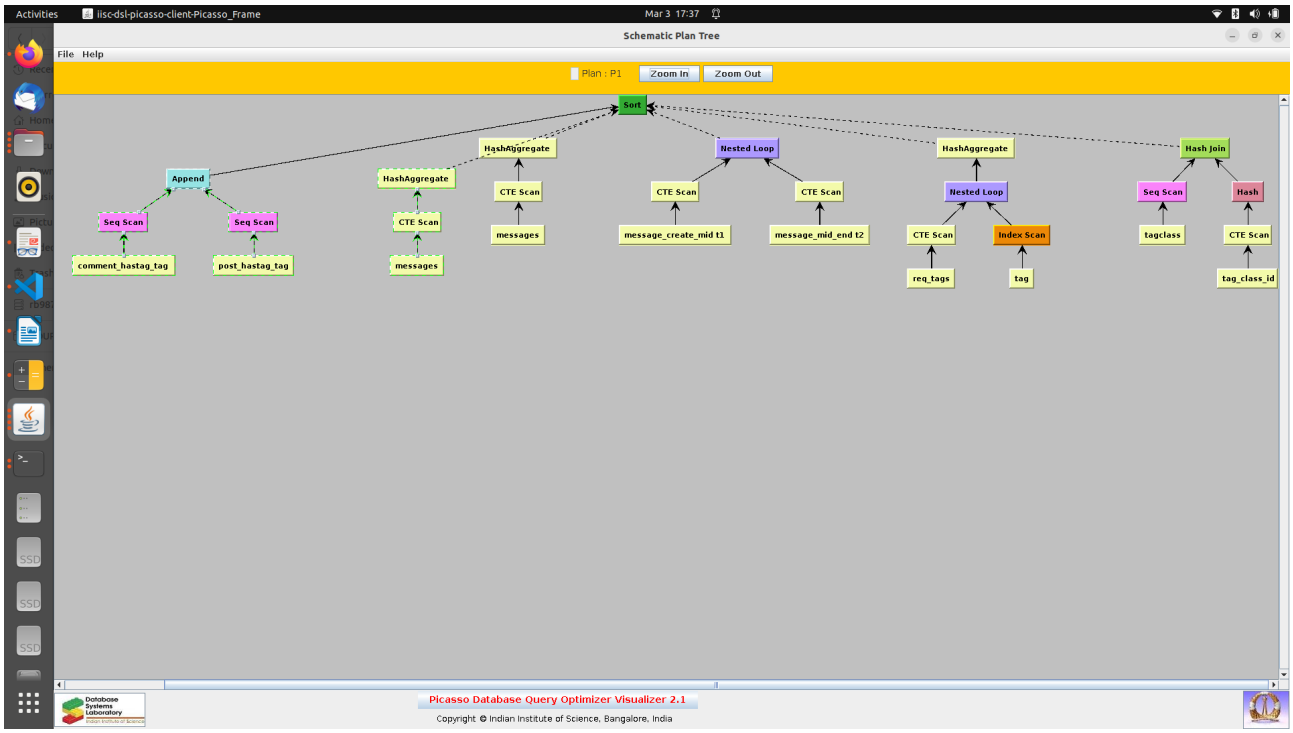
We can see from the plan tree diagram



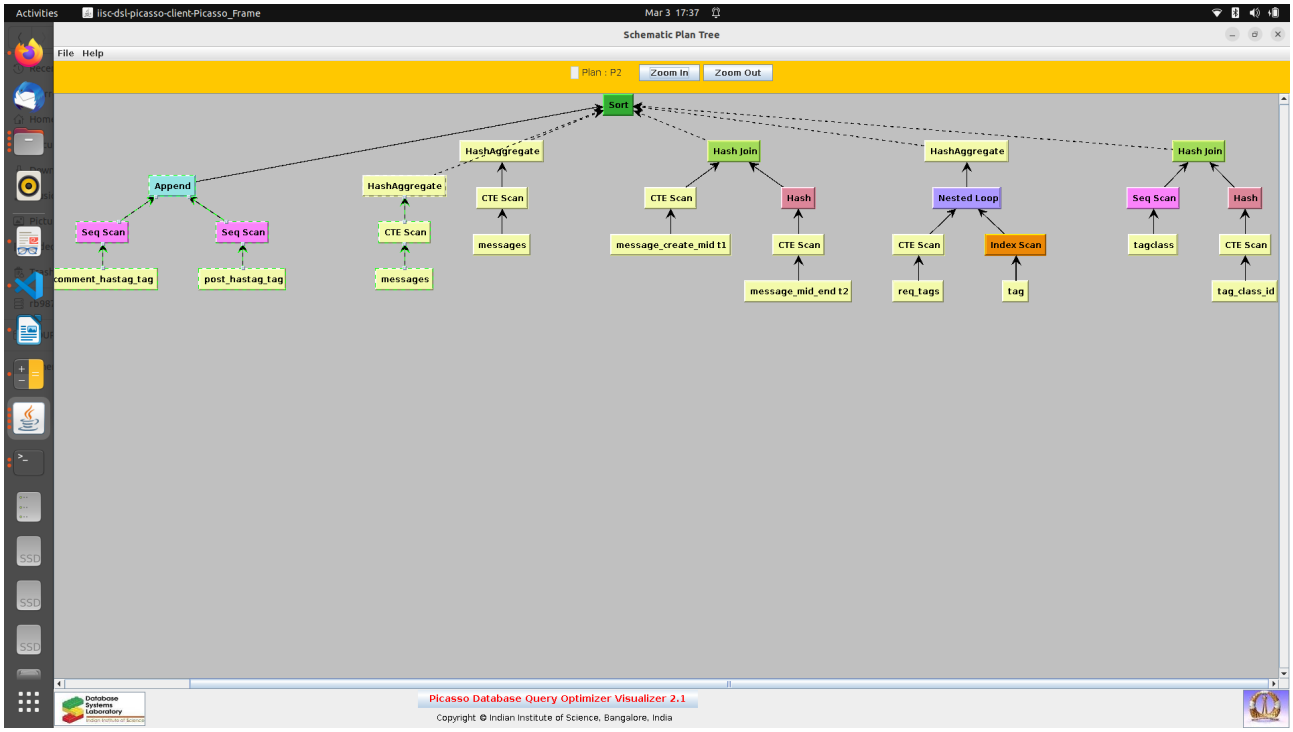
For plan1 and plan2, The query is optimised using nested loop and hash join respecitively. The optimiser finds out which procedure will be more beneficial for the joining and chooses that depending on the number of comments (as messages) and posts (filtered via creation date in chunks(0 – x%).

Plan Trees

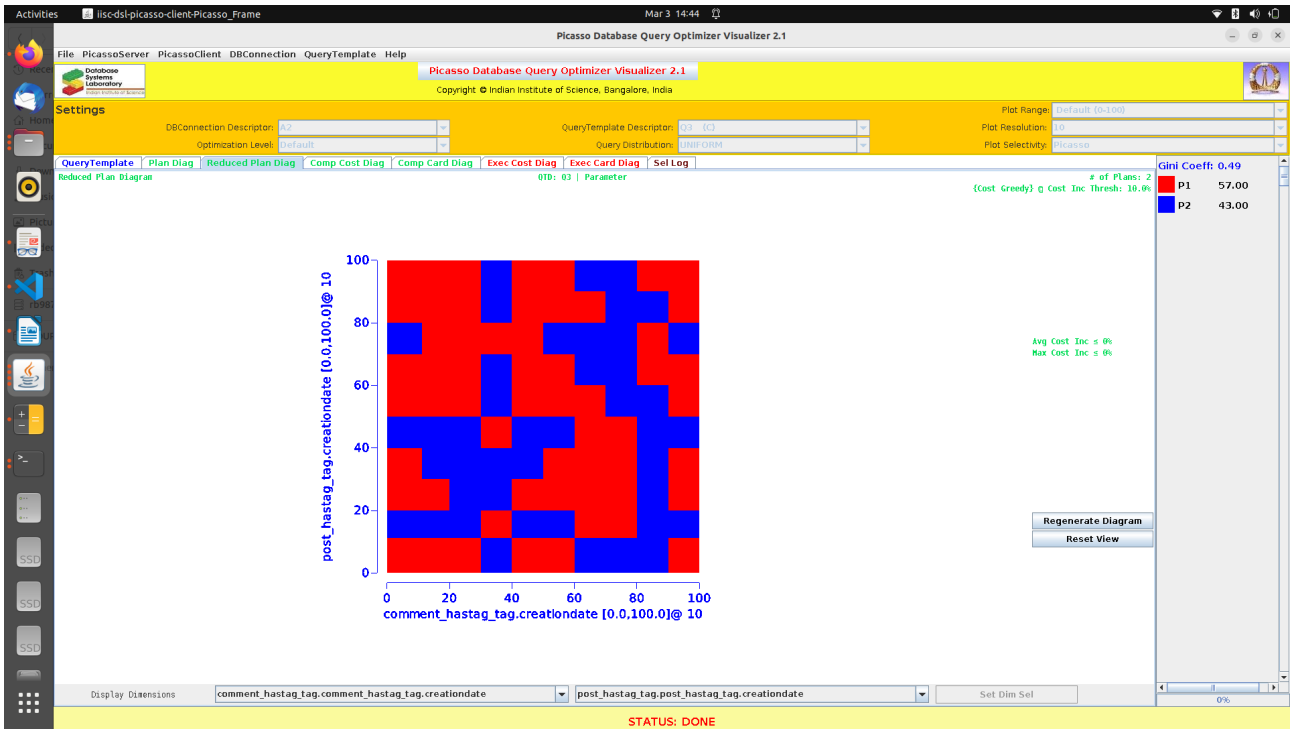
P1:



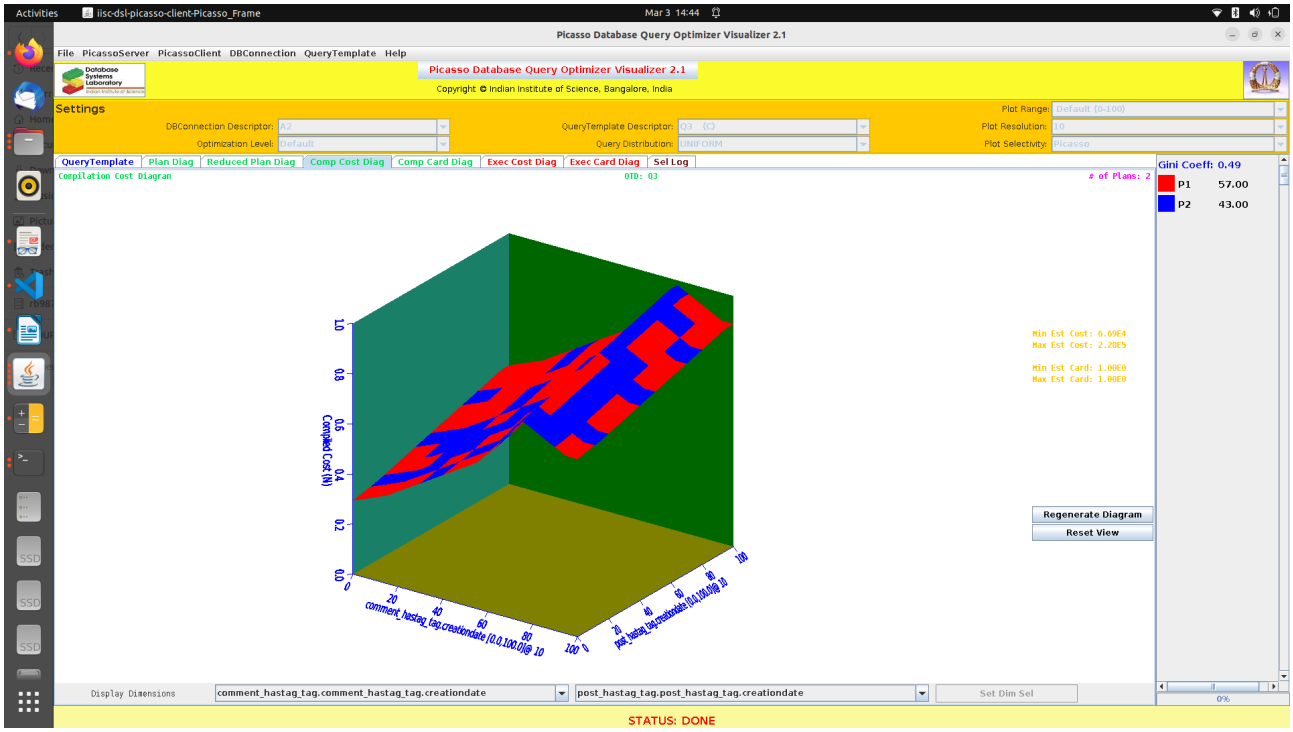
P2:



Reduced Plan Diagram



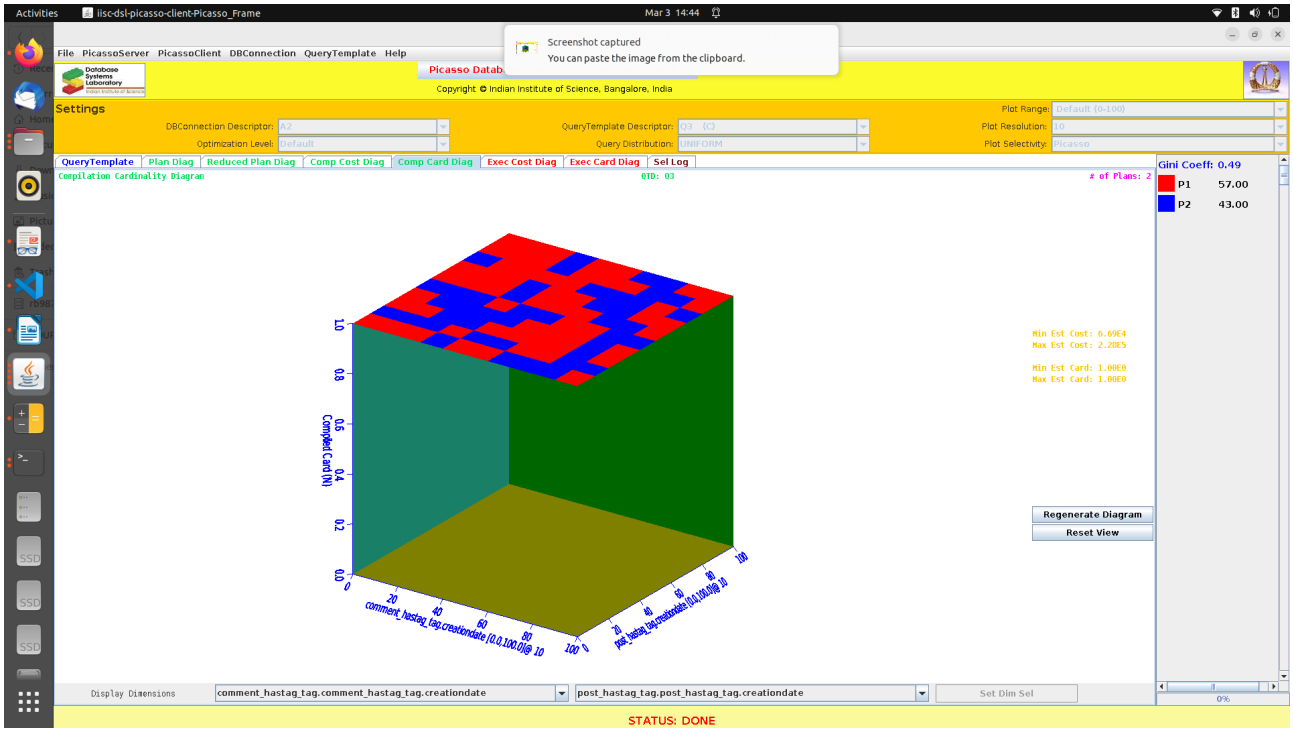
Compiled Cost Diagram



With the increase in the size of chunk (0 – x%) of creation date of the comments_hashtag, The messages in the range of middate to enddate increase, therefore the number of messages having tagcount in the first half , 5 times more , then the messages in the second half.

Therefore, we need to deal with lesser number of messages(and tags) in the next part of the query(tagclasses). Therefore the computational cost decreases.
The increase in the size of chunks of the post_hashtag does not increase the number of posts having tags with creation date in the second range, therefore the compiled time continuously increases with increase in chunk size of post_hashtags

Compiled Cardinality Diagram



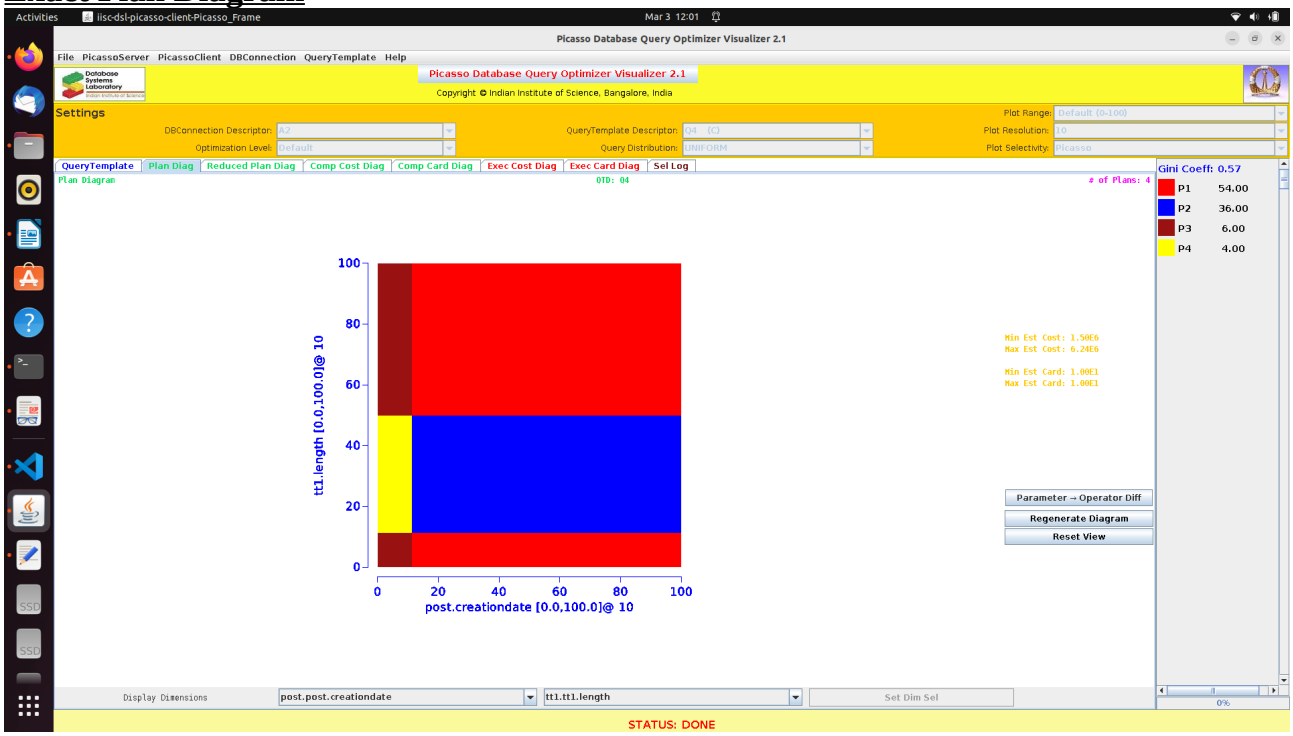
Query – 4

Query Template

```
WITH messagetags AS (
   SELECT post_hastag_tag.postid AS messages, tag.name AS tagname, COUNT(t1.id) AS num_reply
   FROM post_hastag_tag, post, tag, comment t1
   where post.id = post_hastag_tag.postid
   and post_hastag_tag.tagid = tag.id
   and post_hastag_tag.postid = t1.parentpostid
   and post.creationdate :varies
   GROUP BY post_hastag_tag.postid, tag.name
   HAVING COUNT(t1.id) >= 4
UNION ALL
   SELECT comment_hastag_tag.commentid AS messages, tag.name AS tagname, COUNT(tt1.id) AS num_reply
   FROM comment_hastag_tag, comment c1, tag, comment tt1
   WHERE c1.id = comment_hastag_tag.commentid
   and comment_hastag_tag.tagid = tag.id
   and comment_hastag_tag.commentid = tt1.parentcommentid
   and tt1.length :varies
```

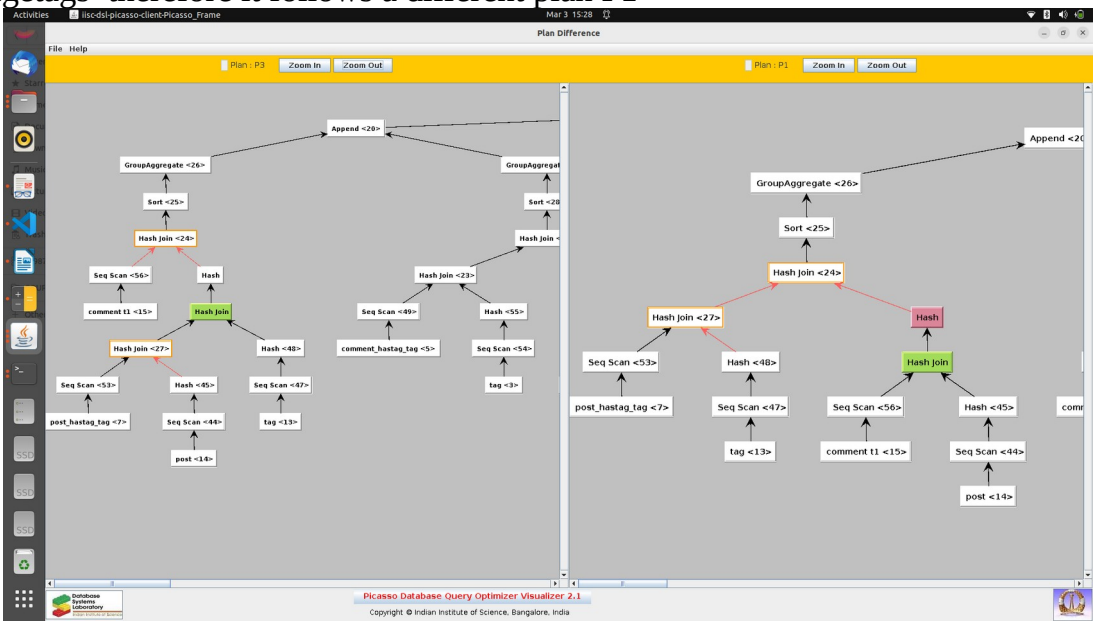
```
    GROUP BY comment_hastag_tag.commentid, tag.name
    HAVING COUNT(tt1.id) >= 4
),
handle0 AS (
    SELECT post_hastag_tag.postid AS messages, tag.name AS tagname, 0 AS num_reply
    FROM post_hastag_tag
    JOIN tag ON post_hastag_tag.tagid = tag.id
UNION ALL
    SELECT comment_hastag_tag.commentid AS messages, tag.name AS tagname, 0 AS num_reply
    FROM comment_hastag_tag
    JOIN tag ON comment_hastag_tag.tagid = tag.id
UNION ALL
    SELECT messages, tagname, num_reply
    FROM messagetags
),
checkreplycount AS (
SELECT messages, tagname, SUM(num_reply)
FROM handle0
GROUP BY messages, tagname
HAVING SUM(num_reply) >= 4
)
SELECT tagname, COUNT(messages)
FROM checkreplycount
GROUP BY tagname
ORDER BY count DESC, tagname
LIMIT 10
```

Exact Plan Diagram



Comments

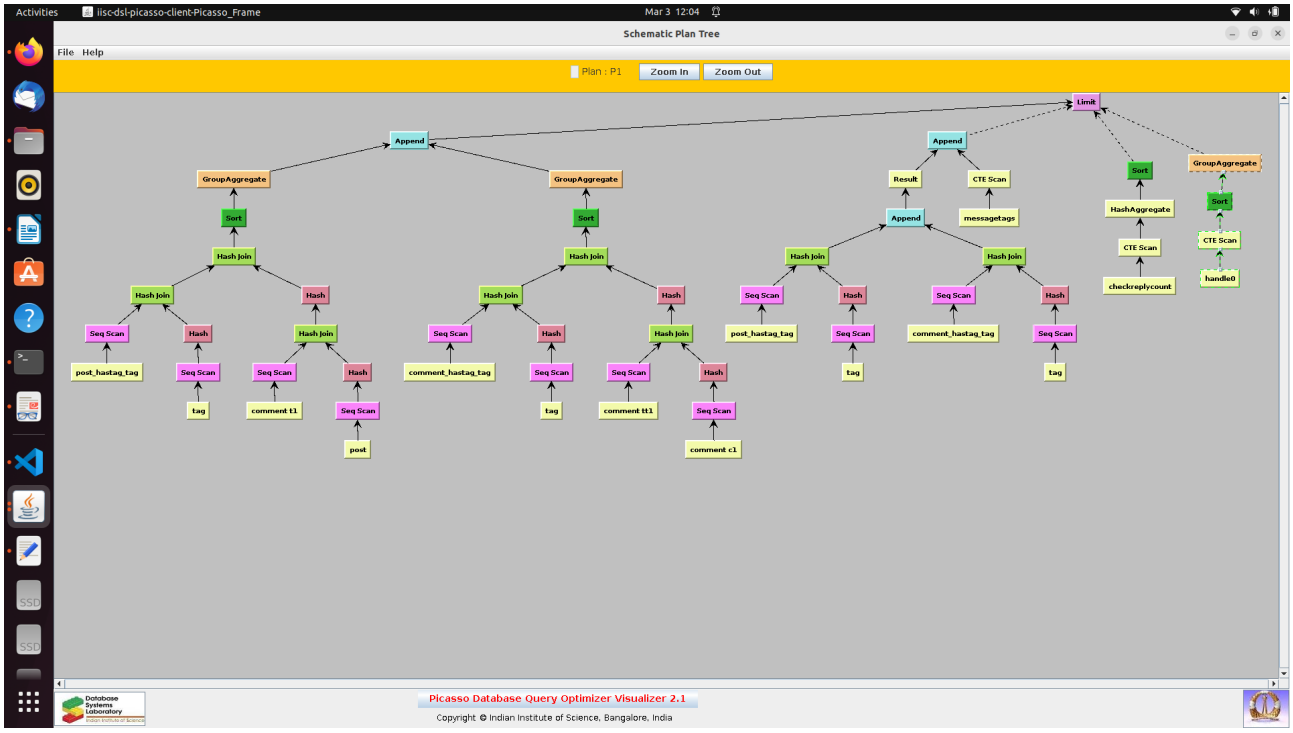
In the first (0 – 10%) chunk of the post (using vary on creationdate), there are very less (or negligible) number of rows in the first term of the union of the with clause of table ‘messagetags’ therefore it follows a different plan P1



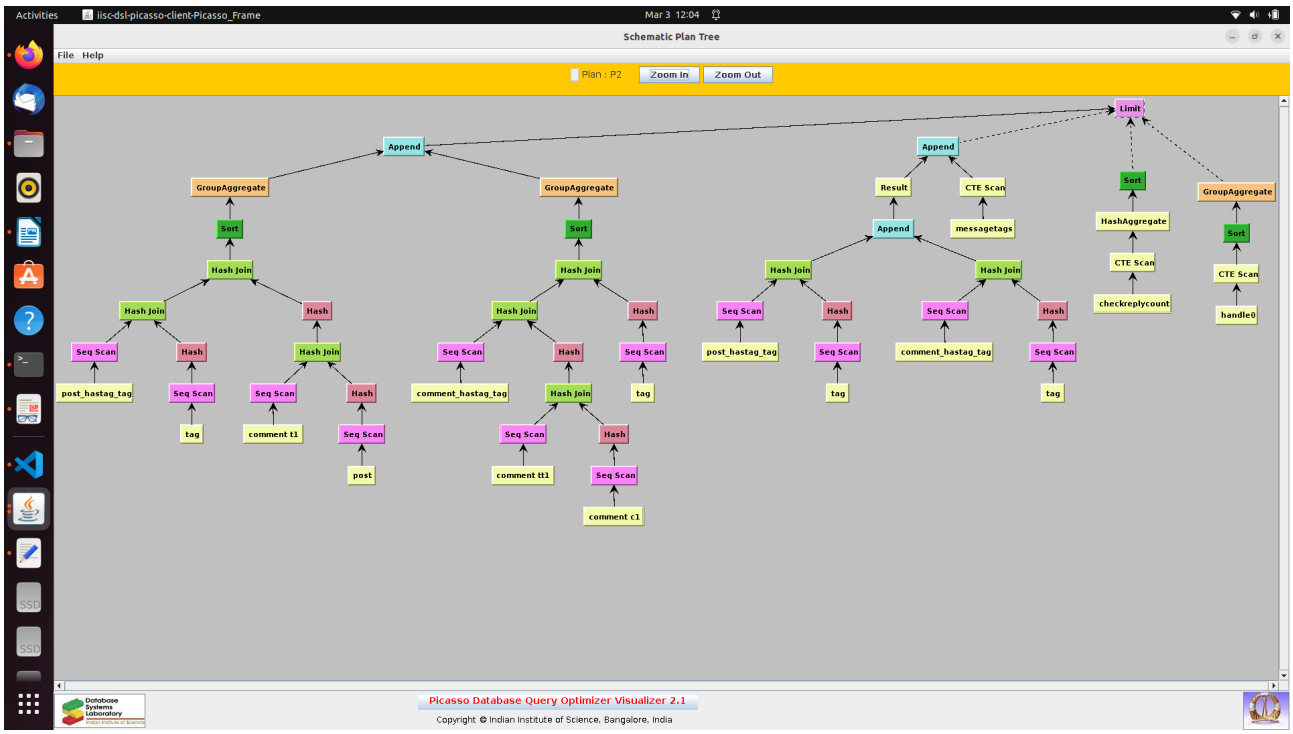
Similarly for the length of comment, there are very less number of rows in the second term of the union of the with clause of the table ‘messagetags’ therefore it follows a different plan P3 or P4. Now as the number increase it need to follow a different strategy for the union

Plan Trees

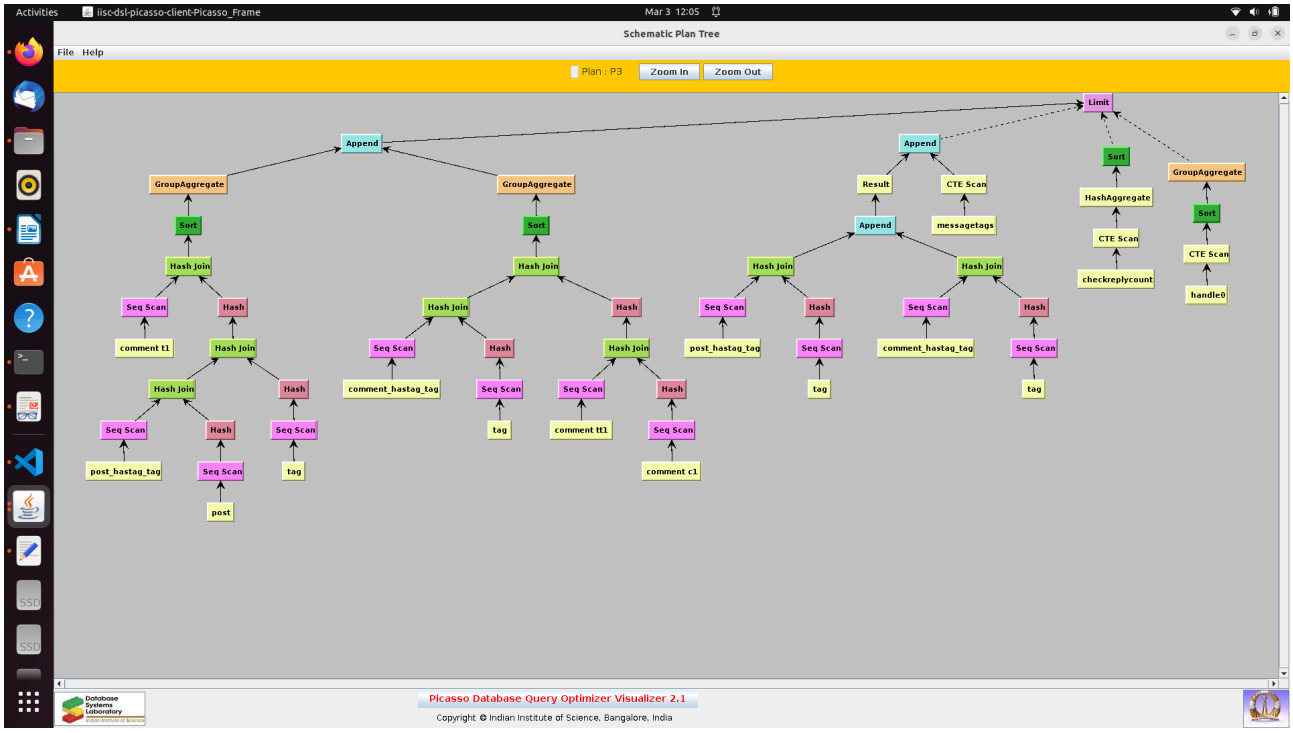
P1:



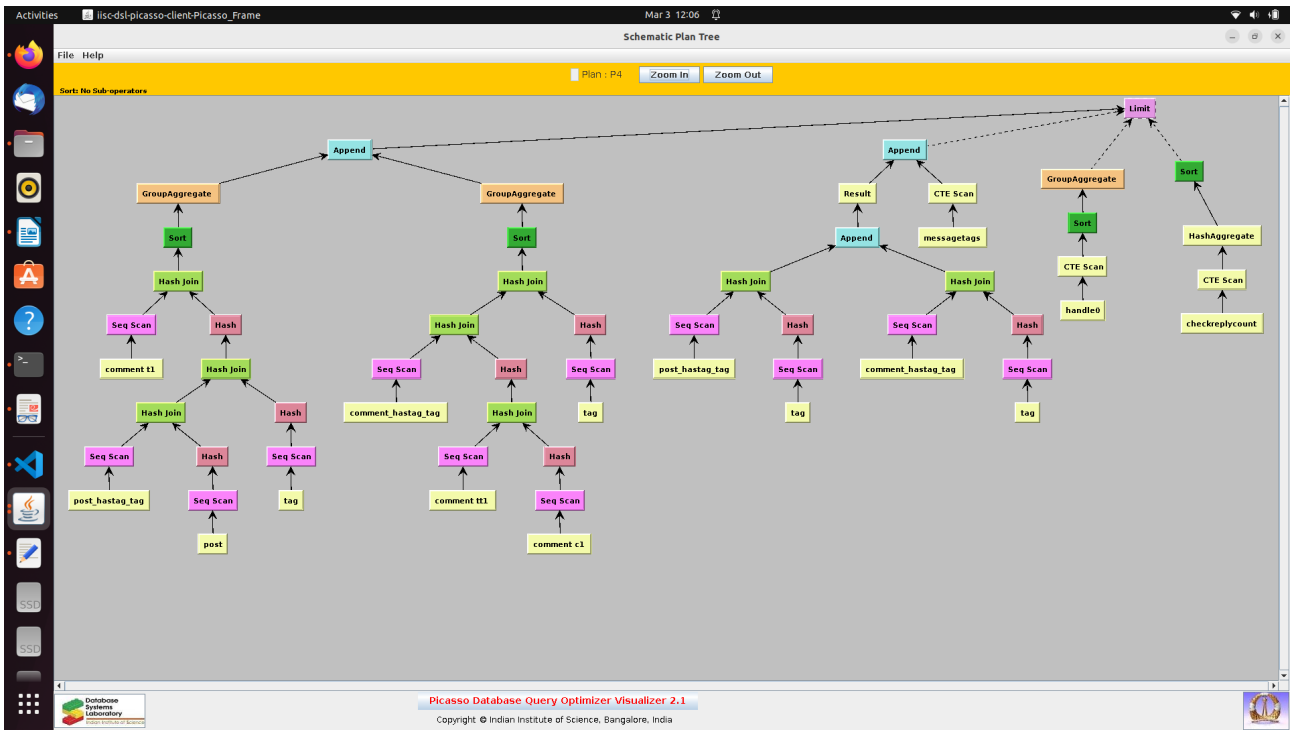
P2:


Activities
iisc-dsl-picasso-client-Picasso_Frame
Mar 3 12:04
Schematic Plan Tree
File Help
Plan : P2
Zoom In
Zoom Out
Limit
Append
GroupAggregate
Sort
Hash Join
Seq Scan
Hash
post_hastag_tag
tag
comment t1
post
comment_hastag_tag
comment tt1
comment c1
Result
CTE Scan
messagetags
comment_hastag_tag
HashAggregate
checkreplycount
handle0
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

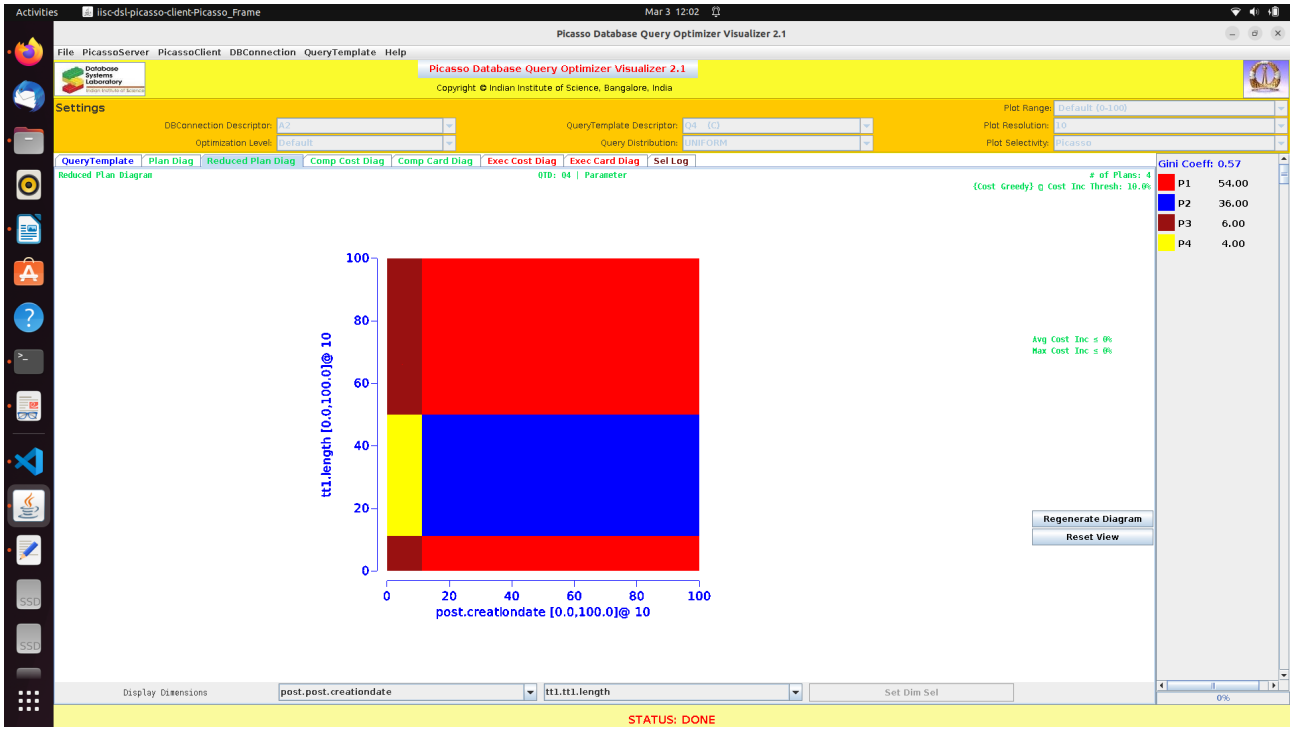
P3:


Activities
iisc-dsl-picasso-client-Picasso_Frame
Mar 3 12:05
Schematic Plan Tree
File Help
Plan : P3
Zoom In
Zoom Out
Limit
Append
GroupAggregate
Sort
Hash Join
Seq Scan
Hash
comment t1
post_hastag_tag
tag
post
comment_hastag_tag
comment tt1
comment c1
Result
CTE Scan
messagetags
HashAggregate
checkreplycount
handle0
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

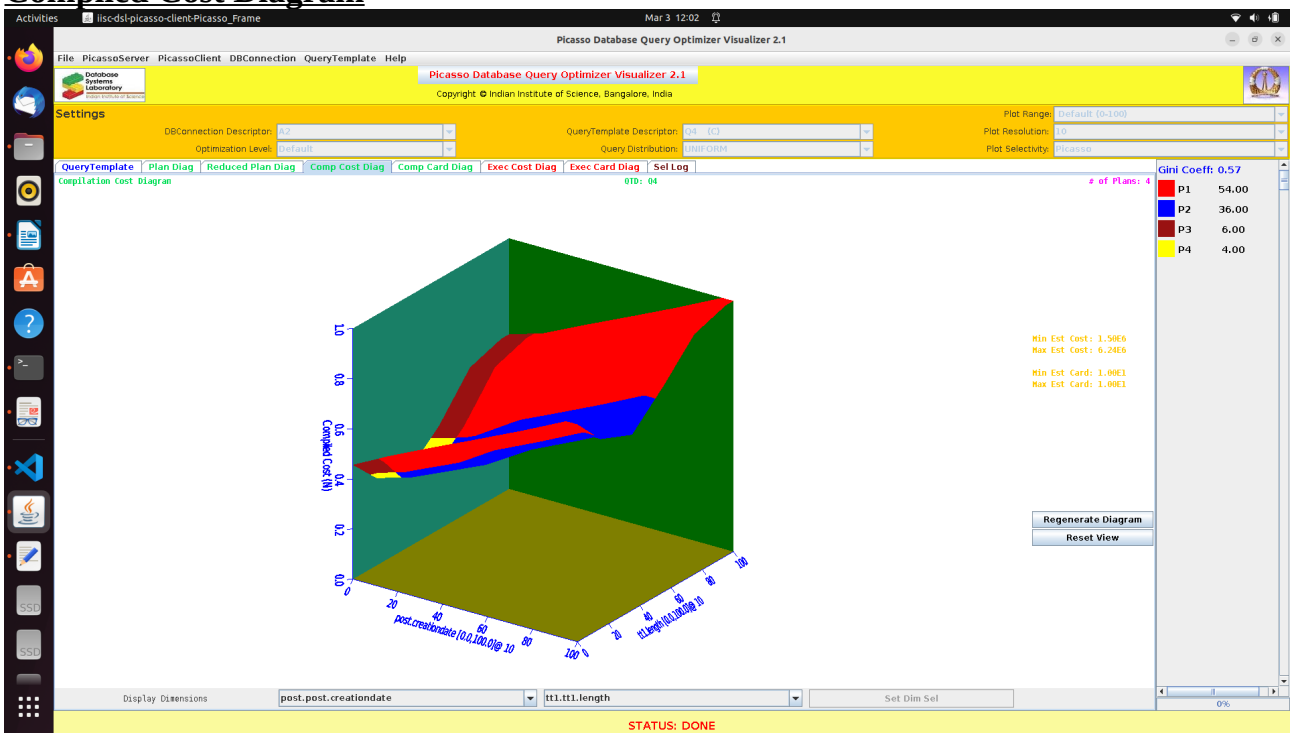
P4:



Reduced Plan Diagram



Compiled Cost Diagram

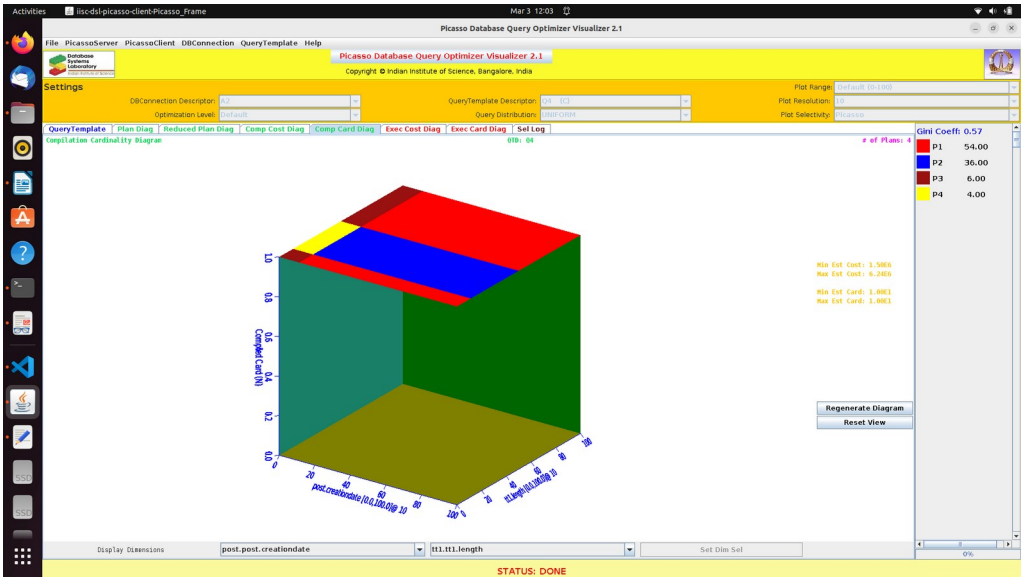


Comments

We can see that the cost keeps on increasing smoothly(almost) with increase in the chunks of post (vary on creationdate), because as we get more and more posts, we will increase the number of rows we have to deal with and will also increase the outputs of the with clause tables.

Now with the increase in the chunk size of the comments(vary on length) we notice that the cost first decrease , this is due to the change in the plan, the optimiser found a better plan and thus the query time is reduced. Now with the increase in the chunks, the time increases because we have to deal with more number of rows and the output of the with – clauses is also larger

Compiled Cardinality Diagram



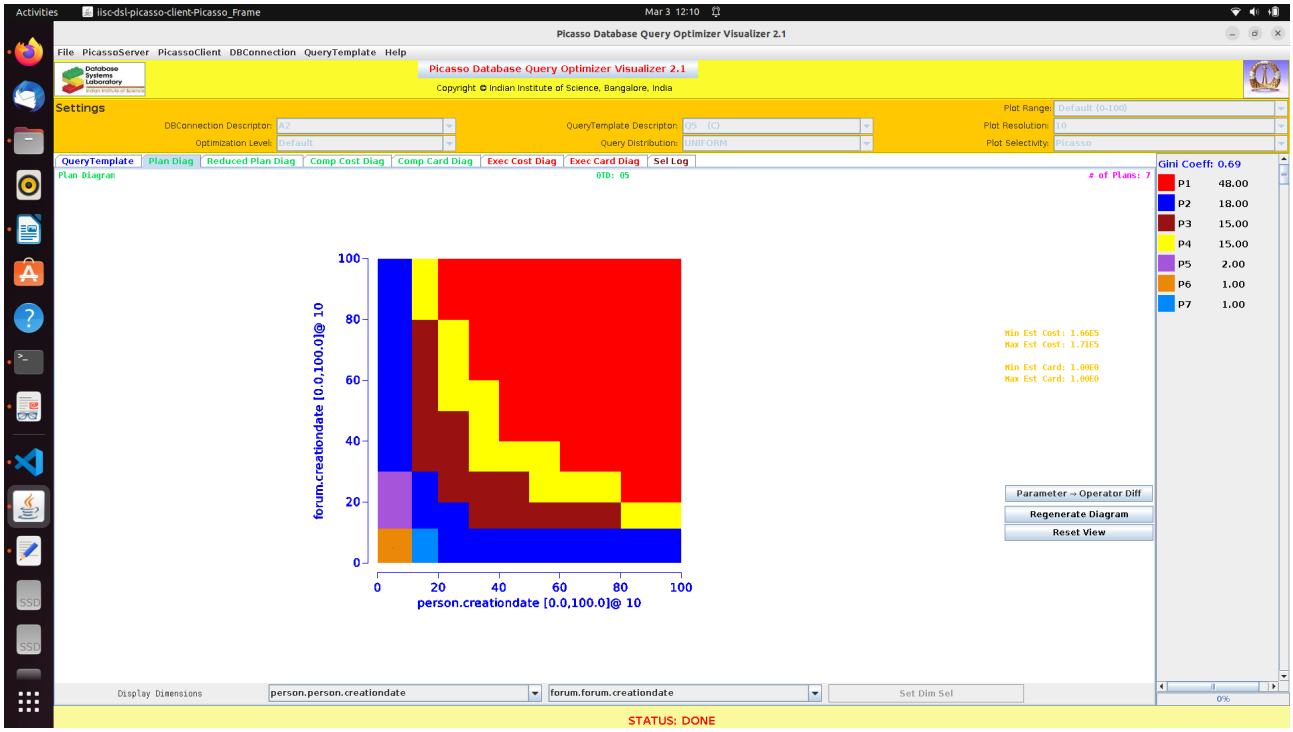
Query – 5

Query Template

```
with req_forum as
(
SELECT forum.id  as forum_id , a.name
FROM forum, person, place a , place b
WHERE forum.ModeratorPersonId = person.id
AND person.creationdate :varies
and  forum.creationdate :varies
AND person.locationcityid = a.id
AND b.name = 'India'
AND a.PartOfPlaceId = b.id
),
req_post_tag as
(
SELECT DISTINCT forum_id
FROM req_forum , Post_hasTag_Tag , Post, Tag , TagClass
WHERE req_forum.forum_id = Post.ContainerForumId
AND Post.id = Post_hasTag_Tag.PostId
AND Post_hasTag_Tag.TagId = Tag.id
AND Tag.TypeTagClassId = tagclass.id
AND tagclass.name = 'TennisPlayer'
),
all_tag as
(
SELECT forum_id , Post_hasTag_Tag.TagId, COUNT(Post_hasTag_Tag.TagId) as tag_count
FROM req_post_tag, Post, Post_hasTag_Tag
WHERE Post.ContainerForumId = req_post_tag.forum_id
AND Post_hasTag_Tag.PostId = Post.id
GROUP BY forum_id , Post_hasTag_Tag.TagId
ORDER BY forum_id
),
max_tag as
(
SELECT forum_id , TagId, tag_count
FROM all_tag
WHERE (forum_id, tag_count) = ANY
    (
    SELECT forum_id , MAX(tag_count)
    FROM all_tag
    GROUP BY forum_id
    )
ORDER BY forum_id
)
SELECT forum_id as forumid, title as forumtitle, tag.name as mostpopulartag,
COUNT(Post_hasTag_Tag.PostId) as count
FROM max_tag, Forum, Post_hasTag_Tag, Post, Tag
WHERE max_tag.forum_id = Forum.id
AND Post.ContainerForumId = max_tag.forum_id
AND Post_hasTag_Tag.PostId = Post.id
```

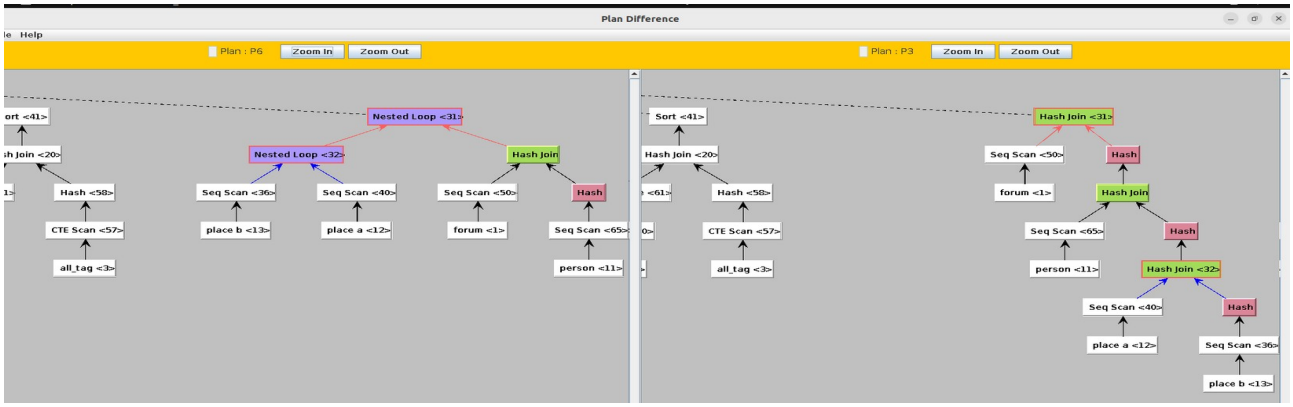
AND Post_hasTag_Tag.TagId = max_tag.TagId
AND Tag.id = max_tag.TagId
GROUP BY forumid , forumtitle, mostpopulartag
ORDER BY count DESC, forumid, forumtitle,mostpopulartag

Exact Plan Diagram



Comments

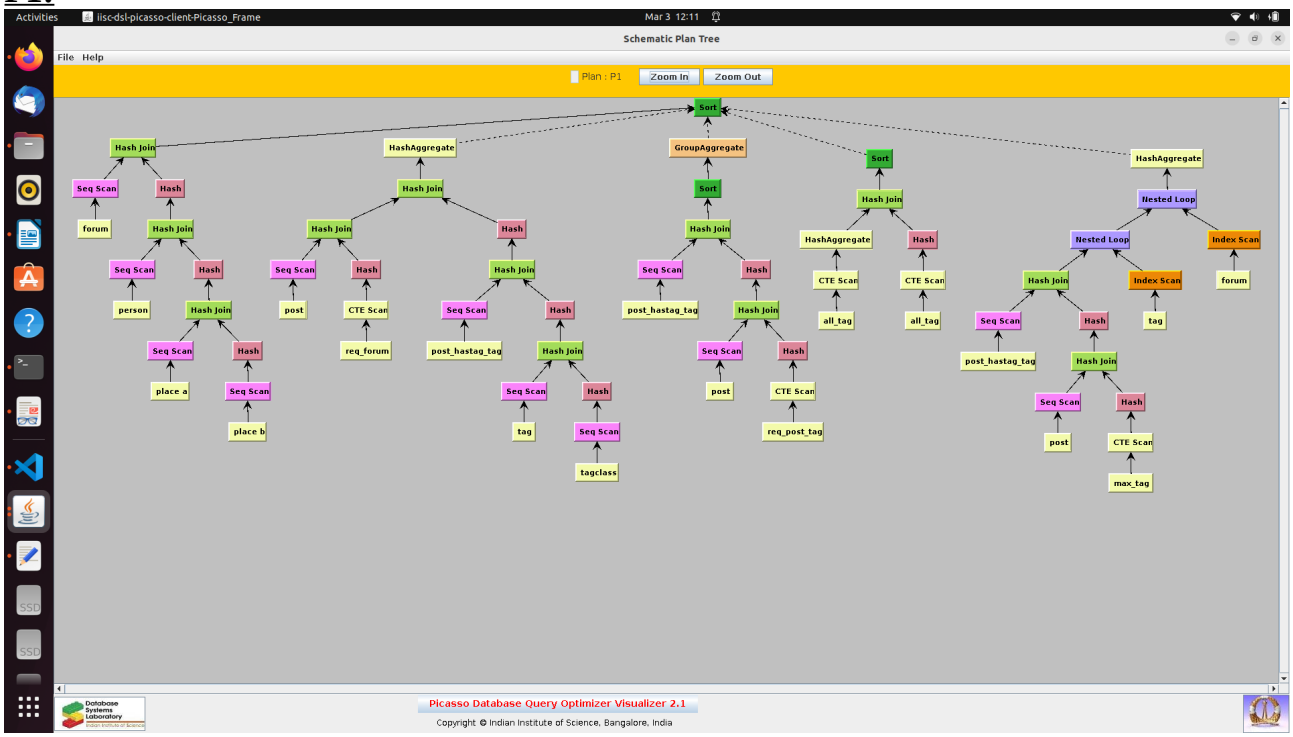
Since we need to join person with forum in the first table, as we increase the chunks (0 – x %) of persons(with vary on creationdate), the number of values that the query has to deal with increases, therefore the optimiser has to change the plans with the increase in number of rows of the person table. This arguments applies to the forum table too
Now since just increasing 1 o



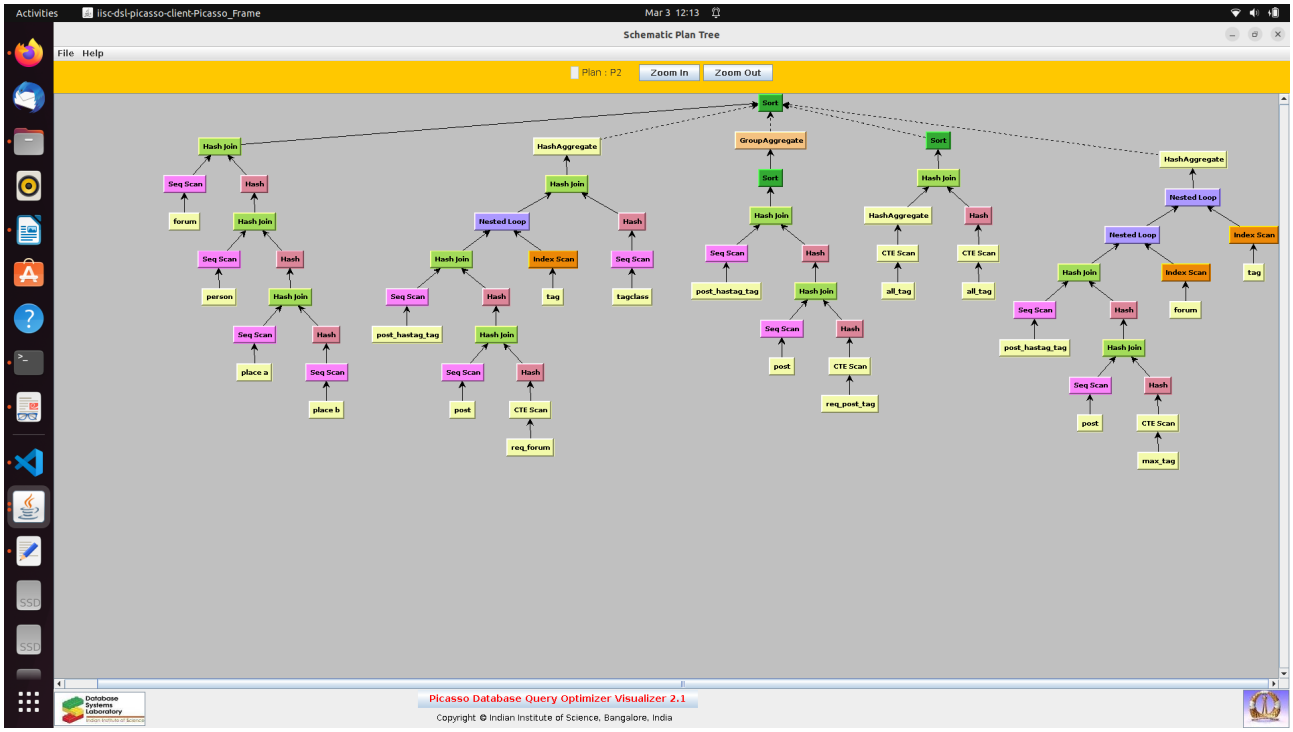
On comparing 2 plans, we can see that the optimiser apply different approach to join the person and forum table

Plan Trees

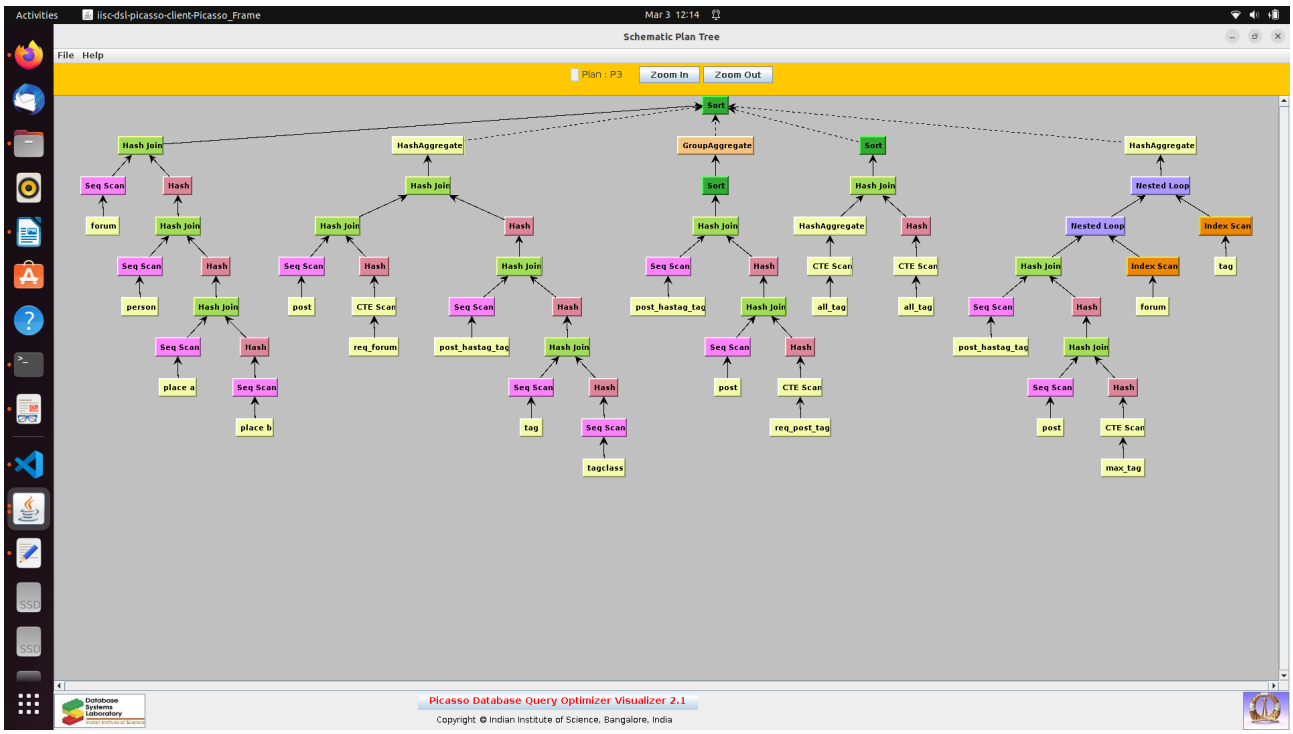
P1:



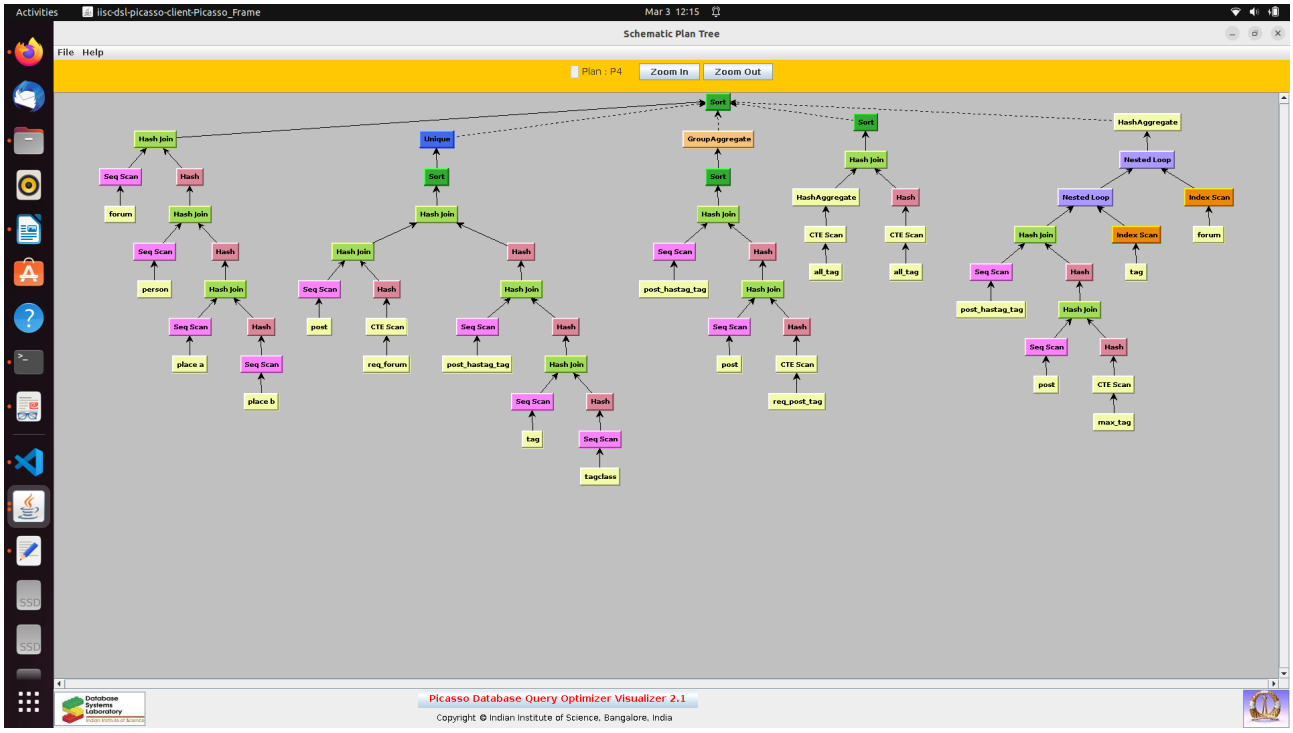
P2:



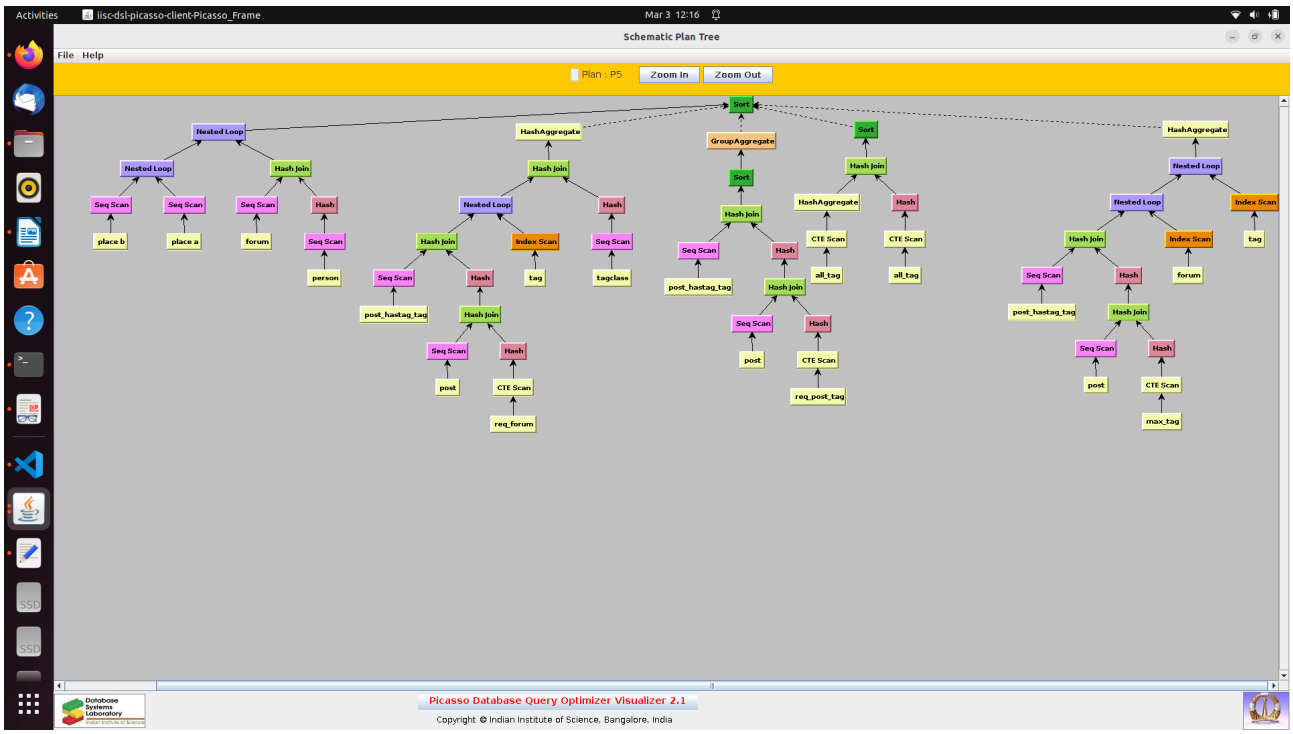
P3:


Activities
iisc-dsl-picasso-client-Picasso_Frame
Mar 3 12:14
Schematic Plan Tree
File Help
Plan : P3
Zoom In
Zoom Out
Sort
Hash Join
Seq Scan
Hash
forum
person
place a
place b
HashAggregate
post
CTE Scan
req_forum
tag
tagclass
GroupAggregate
post_hastag_tag
req_post_tag
all_tag
Nested Loop
Index Scan
post_hastag_tag
max_tag
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

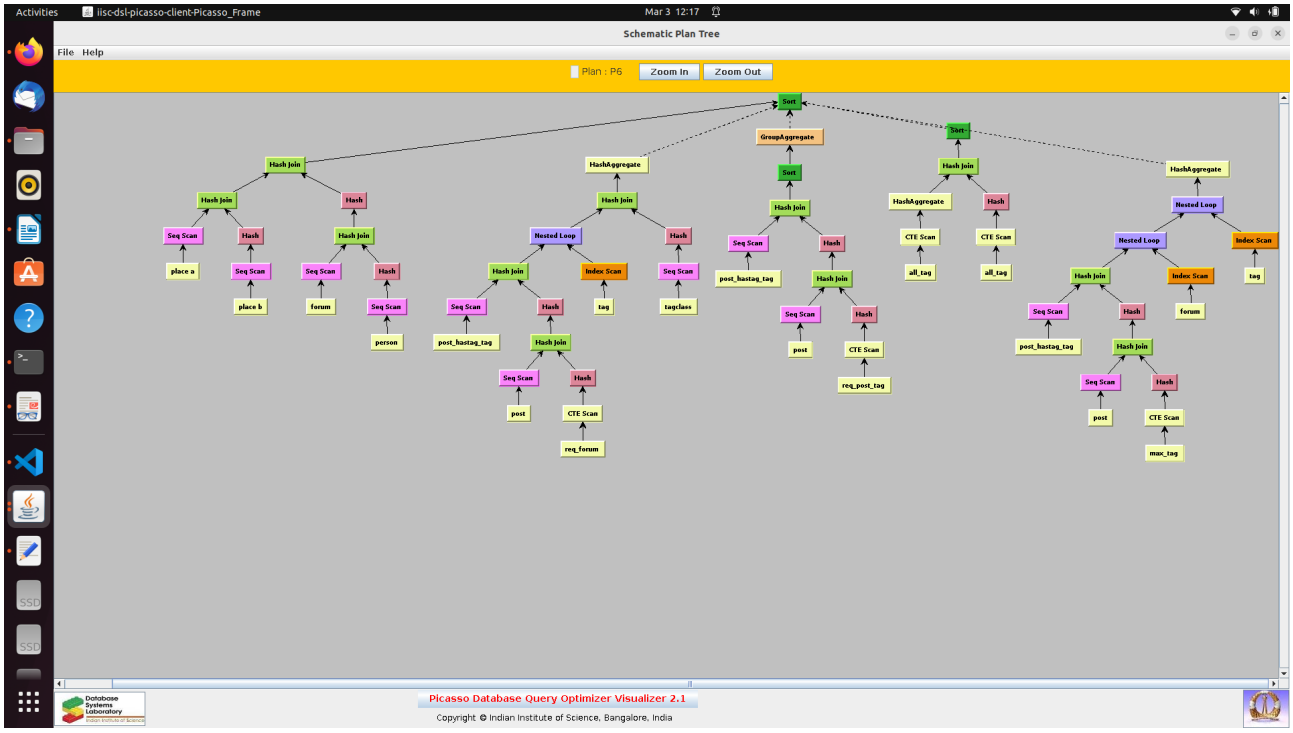
P4:


Activities
iisc-dsl-picasso-client-Picasso_Frame
Mar 3 12:15
Schematic Plan Tree
File Help
Plan : P4
Zoom In
Zoom Out
Unique
Sort
Hash Join
Seq Scan
Hash
forum
person
place a
place b
HashAggregate
post
CTE Scan
req_forum
tag
tagclass
GroupAggregate
post_hastag_tag
req_post_tag
all_tag
Nested Loop
Index Scan
post_hastag_tag
max_tag
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

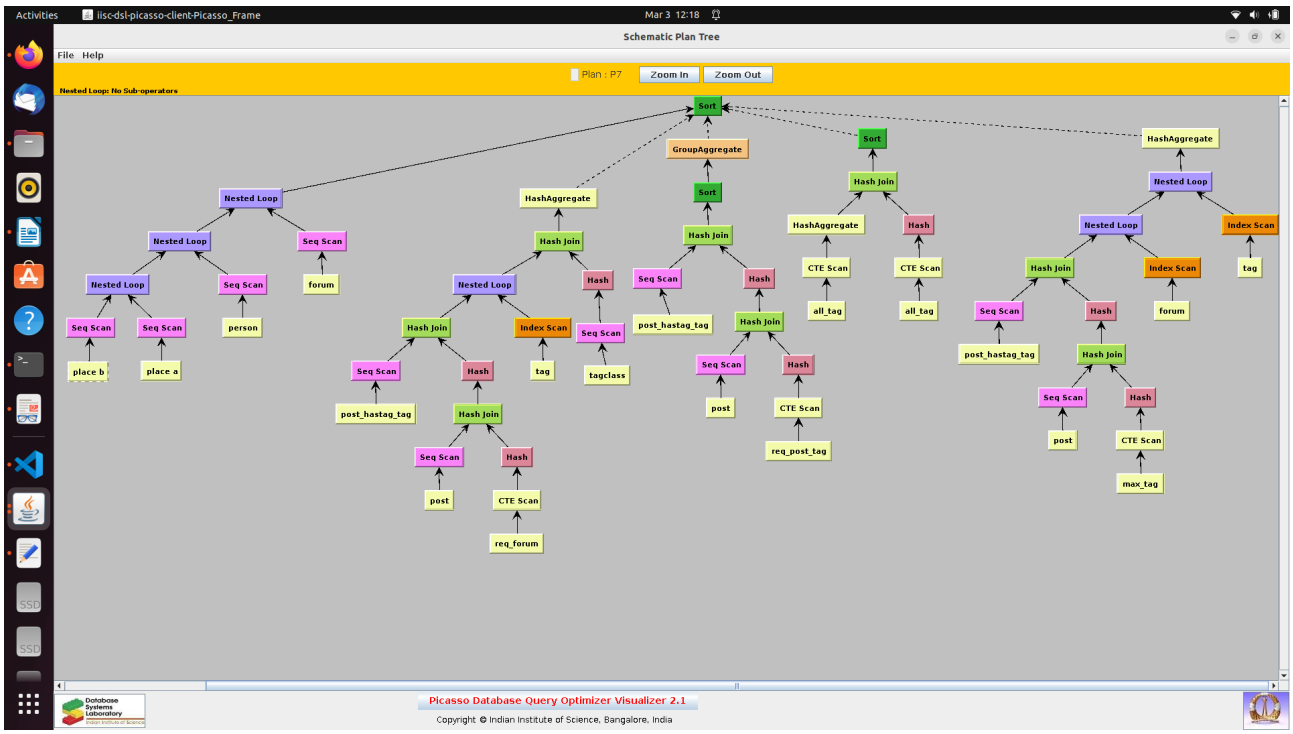
P5:


Activities
iisc-dsl-picasso-client-Picasso_Frame
Mar 3 12:16
Schematic Plan Tree
File Help
Plan : P5
Zoom In
Zoom Out
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

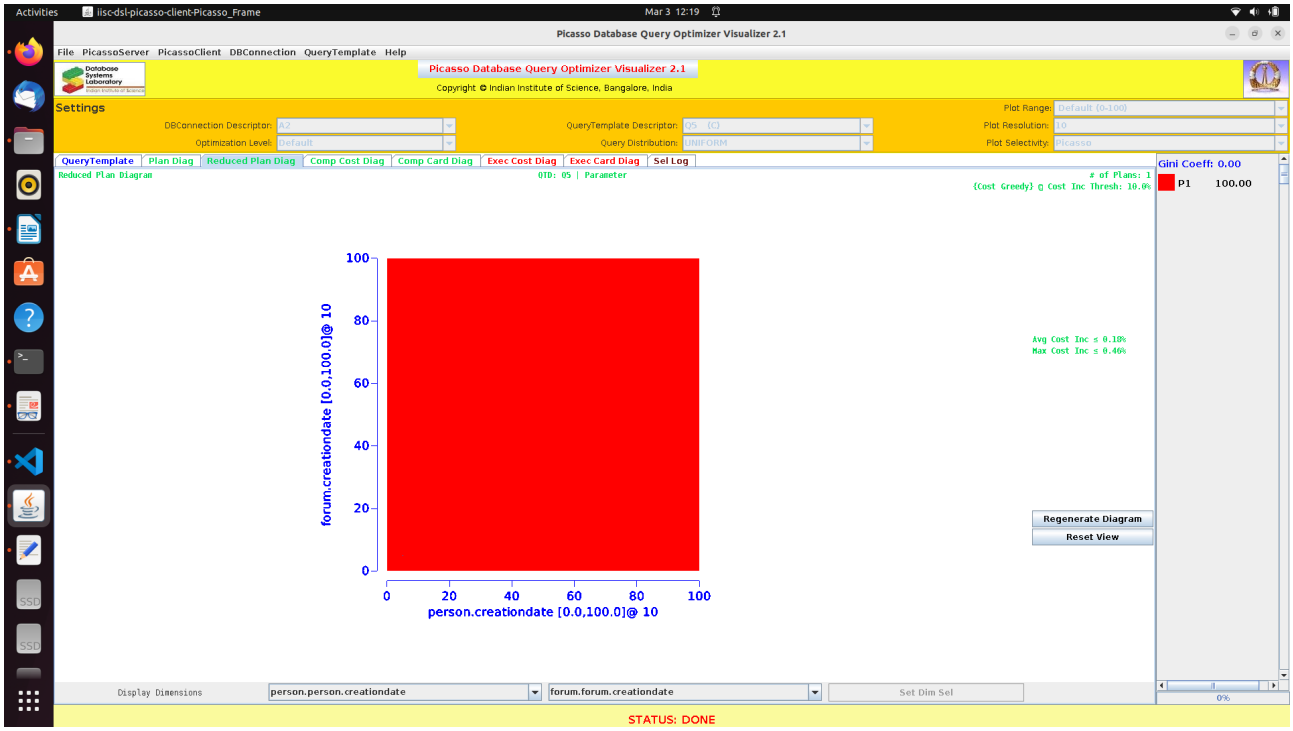
P6:


Activities
iisc-dsl-picasso-client-Picasso_Frame
Mar 3 12:17
Schematic Plan Tree
File Help
Plan : P6
Zoom In
Zoom Out
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India

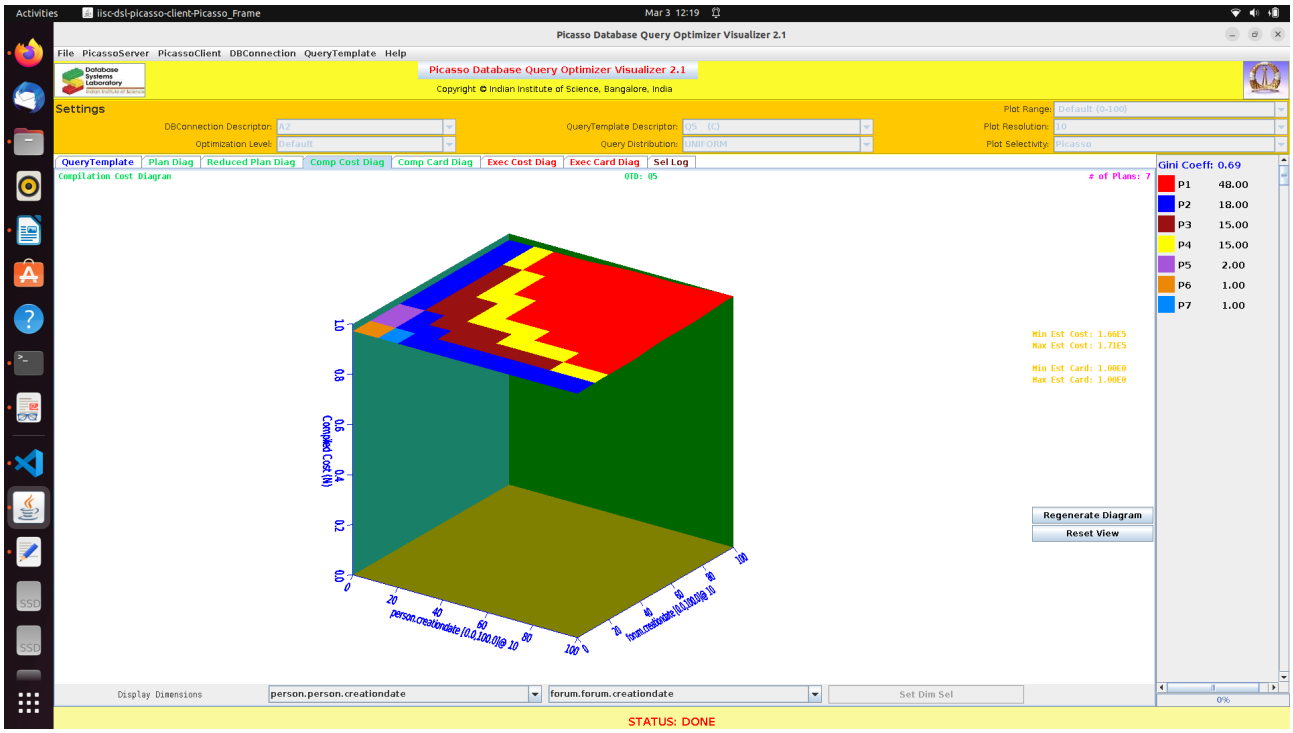
P7:



Reduced Plan Diagram



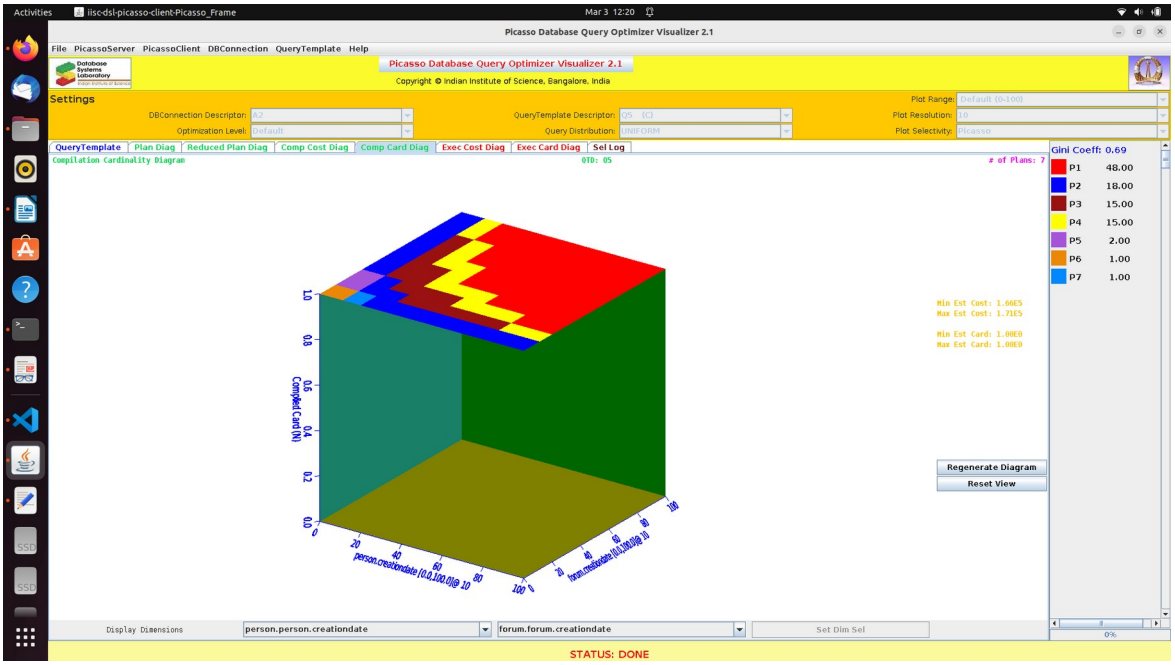
Compiled Cost Diagram



Comments

We can see that the cost remains almost same. This is because we need to do a sequential search on post_hash_tag and place table(cross join) these two operations are independent of the number of rows generated via the varies and these are very heavy operations. Therefore the time is not affected by the varies column much.

Compiled Cardinality Diagram



Here are the .pkt files for your reference -
https://csciitd-my.sharepoint.com/:f:/g/personal/cs5200436_iitd_ac_in/Ep-hwFSYz-JGqT0nFTBfYrsBXYtcDG4KWzbM7JXXJmPTRw?e=avRNaz