Arvados - Bug #14576

[1.3.0] performance: slow postgres collections queries

12/03/2018 10:34 PM - Ward Vandewege

Status:	Resolved	**Start date:**	
Priority:	Normal	**Due date:**	
Assigned To:	Peter Amstutz	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	2018-12-12 Sprint		

Description

Now that qr1hi is on the latest rc for 1.3.0, we're seeing a lot of slow postgres queries like this one:

```
arvados_production=# explain analyze SELECT  collections."uuid", collections."owner_uuid", collect
ions."created_at", collections."modified_by_client_uuid", collections."modified_by_user_uuid", col
lections."modified_at", collections."name", collections."description", collections."properties", c
ollections."portable_data_hash", collections."replication_desired", collections."replication_confi
rmed", collections."replication_confirmed_at", collections."storage_classes_desired", collections.
"storage_classes_confirmed", collections."storage_classes_confirmed_at", collections."delete_at",
collections."trash_at", collections."is_trashed", collections."version", collections."current_vers
ion_uuid", collections."preserve_version" FROM "collections" WHERE (NOT EXISTS(SELECT 1 FROM mater
ialized_permission_view WHERE trashed = 1 AND (collections.owner_uuid = target_uuid)) AND collecti
ons.is_trashed = false AND collections.uuid = collections.current_version_uuid)  ORDER BY collecti
ons.modified_at desc, collections.uuid LIMIT 8 OFFSET 0;
                                                                                                QUERY PLA
N
------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
 Limit  (cost=5470.02..5470.02 rows=1 width=365) (actual time=2533.780..2533.784 rows=8 loops=1)
   ->  Sort  (cost=5470.02..5470.02 rows=1 width=365) (actual time=2533.778..2533.779 rows=8 loops
=1)
         Sort Key: collections.modified_at DESC, collections.uuid
         Sort Method: top-N heapsort  Memory: 29kB
         ->  Merge Anti Join  (cost=5450.80..5470.01 rows=1 width=365) (actual time=1878.913..2399
.362 rows=205414 loops=1)
               Merge Cond: ((collections.owner_uuid)::text = (materialized_permission_view.target_
uuid)::text)
               ->  Sort  (cost=3323.06..3325.62 rows=1027 width=365) (actual time=1878.715..2227.8
90 rows=205414 loops=1)
                     Sort Key: collections.owner_uuid
                     Sort Method: external merge  Disk: 53680kB
                     ->  Bitmap Heap Scan on collections  (cost=53.81..3271.69 rows=1027 width=365
) (actual time=53.099..341.265 rows=205414 loops=1)
                           Recheck Cond: ((NOT is_trashed) AND ((current_version_uuid)::text = (uu
id)::text))
                           Heap Blocks: exact=14972
                           ->  Bitmap Index Scan on index_collections_on_owner_uuid_and_name  (cos
t=0.00..53.55 rows=1027 width=0) (actual time=48.443..48.443 rows=205415 loops=1)
               ->  Sort  (cost=2127.72..2131.33 rows=1447 width=28) (actual time=0.191..0.192 rows
=12 loops=1)
                     Sort Key: materialized_permission_view.target_uuid
                     Sort Method: quicksort  Memory: 25kB
                     ->  Index Only Scan using permission_target_trashed on materialized_permissio
n_view  (cost=0.42..2051.76 rows=1447 width=28) (actual time=0.139..0.156 rows=12 loops=1)
                           Index Cond: (trashed = 1)
                           Heap Fetches: 12
 Planning time: 3.400 ms
 Execution time: 2547.024 ms
(21 rows)
```

Associated revisions

Revision 79bc7fbc - 12/04/2018 03:53 PM - Peter Amstutz

14576: Adjust query for admins that filters out trashed items

refs #14576

Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

Revision 0b2ea830 - 12/04/2018 04:32 PM - Peter Amstutz

Merge branch '14576-query-perf' refs #14576

Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

History

#1 - 12/03/2018 10:35 PM - Ward Vandewege

- Description updated

#2 - 12/03/2018 10:39 PM - Peter Amstutz

Maybe you need to reanalyze the tables to update the planning statistics?

#3 - 12/03/2018 10:40 PM - Peter Amstutz

The "current_version_uuid" check is new in this version.

#4 - 12/03/2018 11:51 PM - Tom Morris

It doesn't appear, at least on the surface, that `current_version_uuid` is the root of the problem. The slow part appears to be resolving this portion of the query:

```
(NOT EXISTS(SELECT 1 FROM materialized_permission_view WHERE trashed = 1 AND (collections.owner_uuid = target_
uuid))
```

A few things that caught my eye:

- it's using a merge sort to resolve the join
- the planner thinks it's going to have 1K rows to sort and it actually gets 205K
- the sort is spilling from memory and being done using disk (50MB)

Not part of the problem, but a couple of other things that caught my eye:

- collections.current_version.uuid should probably be varying(255) instead of varying to match all the other UUID column definitions
- it seems odd that the query is sorting on both modify date and UUID. I can't imagine what value sorting on UUID is adding.

#5 - 12/04/2018 01:26 AM - Ward Vandewege

Bitmap heap scans are used when there are too many results for an index scan (but not so many that a seq scan would be faster).

On 4xphq, there are far fewer results, and the index is used:

```
arvados_production=# explain analyze SELECT  collections."uuid", collections."owner_uuid", collections."create
d_at", collections."modified_by_client_uuid", collections."modified_by_user_uuid", collections."modified_at", 
collections."name", collections."description", collections."properties", collections."portable_data_hash", col
lections."replication_desired", collections."replication_confirmed", collections."replication_confirmed_at", c
ollections."storage_classes_desired", collections."storage_classes_confirmed", collections."storage_classes_co
nfirmed_at", collections."delete_at", collections."trash_at", collections."is_trashed", collections."version",
 collections."current_version_uuid", collections."preserve_version" FROM "collections" WHERE (NOT EXISTS(SELEC
T 1 FROM materialized_permission_view WHERE trashed = 1 AND (collections.owner_uuid = target_uuid)) AND collec
tions.is_trashed = false AND collections.uuid = collections.current_version_uuid)  ORDER BY collections.modifi
ed_at desc, collections.uuid LIMIT 8 OFFSET 0;
)  ORDER BY collections.modified_at desc, collections.uuid LIMIT 8 OFFSET 0;uuid)
                                                                                 QUERY PLAN

---------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------
 Limit  (cost=0.70..529.11 rows=8 width=334) (actual time=0.114..0.191 rows=8 loops=1)
   ->  Nested Loop Anti Join  (cost=0.70..104163.55 rows=1577 width=334) (actual time=0.113..0.186 rows=8 loop
s=1)
         Join Filter: ((collections.owner_uuid)::text = (materialized_permission_view.target_uuid)::text)
         Rows Removed by Join Filter: 56
         ->  Index Scan using index_collections_on_modified_at_uuid on collections  (cost=0.42..103970.30 rows
=1955 width=334) (actual time=0.047..0.097 rows=8 loops=1)
               Filter: ((NOT is_trashed) AND ((uuid)::text = (current_version_uuid)::text))
         ->  Materialize  (cost=0.28..17.32 rows=6 width=28) (actual time=0.003..0.009 rows=7 loops=8)
               ->  Index Only Scan using permission_target_trashed on materialized_permission_view  (cost=0.28
```

```
..17.29 rows=6 width=28) (actual time=0.020..0.054 rows=7 loops=1)
                      Index Cond: (trashed = 1)
                      Heap Fetches: 7
 Planning time: 0.524 ms
 Execution time: 0.236 ms
(12 rows)
```

#6 - 12/04/2018 01:33 AM - Ward Vandewege

Tweaking work_mem makes a significant difference, but I had to dial it up quite high (128MB).

```
arvados_production=# explain (analyze,buffers) SELECT  collections."uuid", collections."owner_uuid", collectio
ns."created_at", collections."modified_by_client_uuid", collections."modified_by_user_uuid", collections."modi
fied_at", collections."name", collections."description", collections."properties", collections."portable_data_
hash", collections."replication_desired", collections."replication_confirmed", collections."replication_confir
med_at", collections."storage_classes_desired", collections."storage_classes_confirmed", collections."storage_
classes_confirmed_at", collections."delete_at", collections."trash_at", collections."is_trashed", collections.
"version", collections."current_version_uuid", collections."preserve_version" FROM "collections" WHERE (NOT EX
ISTS(SELECT 1 FROM materialized_permission_view WHERE trashed = 1 AND (collections.owner_uuid = target_uuid))
AND collections.is_trashed = false AND collections.uuid = collections.current_version_uuid)  ORDER BY collecti
ons.modified_at desc, collections.uuid LIMIT 8 OFFSET 0;
                                                                                                        QUERY PLAN

------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------
 Limit  (cost=5470.11..5470.11 rows=1 width=389) (actual time=2348.268..2348.273 rows=8 loops=1)
   Buffers: shared hit=2574 read=14710, temp read=6722 written=6722
   ->  Sort  (cost=5470.11..5470.11 rows=1 width=389) (actual time=2348.268..2348.269 rows=8 loops=1)
         Sort Key: collections.modified_at DESC, collections.uuid
         Sort Method: top-N heapsort  Memory: 29kB
         Buffers: shared hit=2574 read=14710, temp read=6722 written=6722
         ->  Merge Anti Join  (cost=5450.80..5470.10 rows=1 width=389) (actual time=1719.646..2215.259 rows=20
5411 loops=1)
               Merge Cond: ((collections.owner_uuid)::text = (materialized_permission_view.target_uuid)::text)
               Buffers: shared hit=2574 read=14710, temp read=6722 written=6722
               ->  Sort  (cost=3323.06..3325.62 rows=1027 width=389) (actual time=1719.571..2055.654 rows=2054
11 loops=1)
                     Sort Key: collections.owner_uuid
                     Sort Method: external merge  Disk: 53680kB
                     Buffers: shared hit=2559 read=14710, temp read=6722 written=6722
                     ->  Bitmap Heap Scan on collections  (cost=53.81..3271.69 rows=1027 width=389) (actual ti
me=30.260..260.137 rows=205411 loops=1)
                           Recheck Cond: ((NOT is_trashed) AND ((current_version_uuid)::text = (uuid)::text))
                           Heap Blocks: exact=14972
                           Buffers: shared hit=2559 read=14710
                           ->  Bitmap Index Scan on index_collections_on_owner_uuid_and_name  (cost=0.00..53.5
5 rows=1027 width=0) (actual time=26.685..26.685 rows=205415 loops=1)
                                 Buffers: shared hit=2297
               ->  Sort  (cost=2127.72..2131.33 rows=1447 width=28) (actual time=0.067..0.071 rows=12 loops=1)
                     Sort Key: materialized_permission_view.target_uuid
                     Sort Method: quicksort  Memory: 25kB
                     Buffers: shared hit=15
                     ->  Index Only Scan using permission_target_trashed on materialized_permission_view  (cos
t=0.42..2051.76 rows=1447 width=28) (actual time=0.029..0.044 rows=12 loops=1)
                           Index Cond: (trashed = 1)
                           Heap Fetches: 12
                           Buffers: shared hit=15
 Planning time: 1.008 ms
 Execution time: 2359.227 ms
(29 rows)

arvados_production=# show work_mem;
 work_mem
----------
 4MB
(1 row)

arvados_production=# set work_mem to '64MB';
SET
arvados_production=# show work_mem;
 work_mem
----------
 64MB
(1 row)
```

```
arvados_production=# explain (analyze,buffers) SELECT  collections."uuid", collections."owner_uuid", collectio
ns."created_at", collections."modified_by_client_uuid", collections."modified_by_user_uuid", collections."modi
fied_at", collections."name", collections."description", collections."properties", collections."portable_data_
hash", collections."replication_desired", collections."replication_confirmed", collections."replication_confir
med_at", collections."storage_classes_desired", collections."storage_classes_confirmed", collections."storage_
classes_confirmed_at", collections."delete_at", collections."trash_at", collections."is_trashed", collections.
"version", collections."current_version_uuid", collections."preserve_version" FROM "collections" WHERE (NOT EX
ISTS(SELECT 1 FROM materialized_permission_view WHERE trashed = 1 AND (collections.owner_uuid = target_uuid))
AND collections.is_trashed = false AND collections.uuid = collections.current_version_uuid)  ORDER BY collecti
ons.modified_at desc, collections.uuid LIMIT 8 OFFSET 0;
                                                                                               QUERY PLAN

---------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
 Limit  (cost=5470.11..5470.11 rows=1 width=389) (actual time=2834.149..2834.152 rows=8 loops=1)
   Buffers: shared hit=16 read=17268, temp read=6721 written=6721
   ->  Sort  (cost=5470.11..5470.11 rows=1 width=389) (actual time=2834.147..2834.149 rows=8 loops=1)
         Sort Key: collections.modified_at DESC, collections.uuid
         Sort Method: top-N heapsort  Memory: 29kB
         Buffers: shared hit=16 read=17268, temp read=6721 written=6721
         ->  Merge Anti Join  (cost=5450.80..5470.10 rows=1 width=389) (actual time=2321.278..2701.787 rows=20
5411 loops=1)
               Merge Cond: ((collections.owner_uuid)::text = (materialized_permission_view.target_uuid)::text)
               Buffers: shared hit=16 read=17268, temp read=6721 written=6721
               ->  Sort  (cost=3323.06..3325.62 rows=1027 width=389) (actual time=2321.208..2497.855 rows=2054
11 loops=1)
                     Sort Key: collections.owner_uuid
                     Sort Method: external merge  Disk: 53760kB
                     Buffers: shared hit=1 read=17268, temp read=6721 written=6721
                     ->  Bitmap Heap Scan on collections  (cost=53.81..3271.69 rows=1027 width=389) (actual ti
me=37.795..252.841 rows=205411 loops=1)
                           Recheck Cond: ((NOT is_trashed) AND ((current_version_uuid)::text = (uuid)::text))
                           Heap Blocks: exact=14972
                           Buffers: shared hit=1 read=17268
                           ->  Bitmap Index Scan on index_collections_on_owner_uuid_and_name  (cost=0.00..53.5
5 rows=1027 width=0) (actual time=34.232..34.232 rows=205415 loops=1)
                                 Buffers: shared read=2297
               ->  Sort  (cost=2127.72..2131.33 rows=1447 width=28) (actual time=0.062..0.067 rows=12 loops=1)
                     Sort Key: materialized_permission_view.target_uuid
                     Sort Method: quicksort  Memory: 25kB
                     Buffers: shared hit=15
                     ->  Index Only Scan using permission_target_trashed on materialized_permission_view  (cos
t=0.42..2051.76 rows=1447 width=28) (actual time=0.025..0.042 rows=12 loops=1)
                           Index Cond: (trashed = 1)
                           Heap Fetches: 12
                           Buffers: shared hit=15
 Planning time: 1.024 ms
 Execution time: 2848.279 ms
(29 rows)

arvados_production=# set work_mem to '128MB';
SET
arvados_production=# show work_mem;
 work_mem
----------
 128MB
(1 row)

arvados_production=# explain (analyze,buffers) SELECT  collections."uuid", collections."owner_uuid", collectio
ns."created_at", collections."modified_by_client_uuid", collections."modified_by_user_uuid", collections."modi
fied_at", collections."name", collections."description", collections."properties", collections."portable_data_
hash", collections."replication_desired", collections."replication_confirmed", collections."replication_confir
med_at", collections."storage_classes_desired", collections."storage_classes_confirmed", collections."storage_
classes_confirmed_at", collections."delete_at", collections."trash_at", collections."is_trashed", collections.
"version", collections."current_version_uuid", collections."preserve_version" FROM "collections" WHERE (NOT EX
ISTS(SELECT 1 FROM materialized_permission_view WHERE trashed = 1 AND (collections.owner_uuid = target_uuid))
AND collections.is_trashed = false AND collections.uuid = collections.current_version_uuid)  ORDER BY collecti
ons.modified_at desc, collections.uuid LIMIT 8 OFFSET 0;
                                                                                               QUERY PLAN

---------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
 Limit  (cost=5470.11..5470.11 rows=1 width=389) (actual time=1083.613..1083.618 rows=8 loops=1)
   Buffers: shared hit=17 read=17267
   ->  Sort  (cost=5470.11..5470.11 rows=1 width=389) (actual time=1083.610..1083.612 rows=8 loops=1)
```

```
        Sort Key: collections.modified_at DESC, collections.uuid
        Sort Method: top-N heapsort  Memory: 29kB
        Buffers: shared hit=17 read=17267
        ->  Merge Anti Join  (cost=5450.80..5470.10 rows=1 width=389) (actual time=693.560..946.680 rows=2054
11 loops=1)
              Merge Cond: ((collections.owner_uuid)::text = (materialized_permission_view.target_uuid)::text)
              Buffers: shared hit=17 read=17267
              ->  Sort  (cost=3323.06..3325.62 rows=1027 width=389) (actual time=693.492..755.000 rows=205411
 loops=1)
                    Sort Key: collections.owner_uuid
                    Sort Method: quicksort  Memory: 97787kB
                    Buffers: shared hit=2 read=17267
                    ->  Bitmap Heap Scan on collections  (cost=53.81..3271.69 rows=1027 width=389) (actual ti
me=37.848..245.194 rows=205411 loops=1)
                          Recheck Cond: ((NOT is_trashed) AND ((current_version_uuid)::text = (uuid)::text))
                          Heap Blocks: exact=14972
                          Buffers: shared hit=2 read=17267
                          ->  Bitmap Index Scan on index_collections_on_owner_uuid_and_name  (cost=0.00..53.5
5 rows=1027 width=0) (actual time=34.184..34.184 rows=205415 loops=1)
                                Buffers: shared read=2297
              ->  Sort  (cost=2127.72..2131.33 rows=1447 width=28) (actual time=0.060..0.062 rows=12 loops=1)
                    Sort Key: materialized_permission_view.target_uuid
                    Sort Method: quicksort  Memory: 25kB
                    Buffers: shared hit=15
                    ->  Index Only Scan using permission_target_trashed on materialized_permission_view  (cos
t=0.42..2051.76 rows=1447 width=28) (actual time=0.024..0.040 rows=12 loops=1)
                          Index Cond: (trashed = 1)
                          Heap Fetches: 12
                          Buffers: shared hit=15
 Planning time: 1.028 ms
 Execution time: 1086.900 ms
(29 rows)
```

work_mem is used for each operation in each session. We are configured for 512 concurrent connections; at 128MB per connection (and per table!) that is too much for this box, for sure...

But it helps a lot, because we no longer have an external merge to disk>

#7 - 12/04/2018 07:15 AM - Tom Morris

Are we sure the stats are accurate? 205K vs 1K is a pretty big mistake for the query planner to make.

Sorting in memory vs spilling to disk when sorting will definitely be faster, but I don't think it should be doing a merge-sort for the join in the first place.

1 sec is way faster then 3 sec, but still not as good as one would hope for a relatively straightforward query.

#8 - 12/04/2018 01:26 PM - Ward Vandewege

Tom Morris wrote:

> Are we sure the stats are accurate? 205K vs 1K is a pretty big mistake for the query planner to make.
>
> Sorting in memory vs spilling to disk when sorting will definitely be faster, but I don't think it should be doing a merge-sort for the join in the first place.
>
> 1 sec is way faster then 3 sec, but still not as good as one would hope for a relatively straightforward query.

I had already run analyze on all tables (we have autovacuum enabled).

I just now ran an explicit vacuum analyze on the collections table. That seems to have helped, the planner now doesn't think it's going to get 1k rows back. It doesn't help with the overall performance of the query though, which remains similar. The query plan now looks like this:

```
                                                                                     QUERY PLAN

-----------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------
 Limit  (cost=40451.00..40451.00 rows=1 width=368) (actual time=971.355..971.359 rows=8 loops=1)
   ->  Sort  (cost=40451.00..40451.00 rows=1 width=368) (actual time=971.354..971.357 rows=8 loops=1)
        Sort Key: collections.modified_at DESC, collections.uuid
        Sort Method: top-N heapsort  Memory: 29kB
        ->  Merge Anti Join  (cost=37304.34..40450.99 rows=1 width=368) (actual time=599.003..838.943 rows=20
5411 loops=1)
              Merge Cond: ((collections.owner_uuid)::text = (materialized_permission_view.target_uuid)::text)
              ->  Sort  (cost=35175.39..35688.74 rows=205342 width=368) (actual time=598.932..657.342 rows=20
```

```
5411 loops=1)
                     Sort Key: collections.owner_uuid
                     Sort Method: quicksort  Memory: 97787kB
                     ->  Seq Scan on collections  (cost=0.00..17056.35 rows=205342 width=368) (actual time=0.0
06..166.463 rows=205411 loops=1)
                           Filter: (NOT is_trashed)
                           Rows Removed by Filter: 2124
               ->  Sort  (cost=2127.72..2131.33 rows=1447 width=28) (actual time=0.061..0.064 rows=12 loops=1)
                     Sort Key: materialized_permission_view.target_uuid
                     Sort Method: quicksort  Memory: 25kB
                     ->  Index Only Scan using permission_target_trashed on materialized_permission_view  (cos
t=0.42..2051.76 rows=1447 width=28) (actual time=0.024..0.037 rows=12 loops=1)
                           Index Cond: (trashed = 1)
                           Heap Fetches: 12
 Planning time: 1.304 ms
 Execution time: 973.785 ms
(20 rows)
```

#9 - 12/04/2018 02:51 PM - Ward Vandewege

This query originates from app/models/arvados_model.rb around line 277:

```
      # Admin skips most permission checks, but still want to filter on trashed items.
      if !include_trash
        if sql_table != "api_client_authorizations"
          # Only include records where the owner is not trashed
          sql_conds = "NOT EXISTS(SELECT 1 FROM #{PERMISSION_VIEW} "+
                      "WHERE trashed = 1 AND "+
                      "(#{sql_table}.owner_uuid = target_uuid)) #{exclude_trashed_records}"
        end
      end
```

It is an admin-only query.

#10 - 12/04/2018 02:55 PM - Tom Clegg

Adjusting the "is trash?" subquery can make a big difference.

```
EXPLAIN ANALYZE SELECT *
  FROM "collections"
  WHERE 1 IS DISTINCT FROM (SELECT MAX(trashed) FROM materialized_permission_view WHERE collections.owner_uuid
 = target_uuid)
    AND collections.is_trashed = false
  ORDER BY collections.modified_at desc, collections.uuid
  LIMIT 8 OFFSET 0;
                                                                                                     QUERY PLAN

--------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
 Limit  (cost=0.42..715.63 rows=8 width=627) (actual time=0.178..15.753 rows=8 loops=1)
   ->  Index Scan using index_collections_on_modified_at_uuid on collections  (cost=0.42..18277851.14 rows=204
446 width=627) (actual time=0.178..15.751 rows=8 loops=1)
         Filter: ((NOT is_trashed) AND (1 IS DISTINCT FROM (SubPlan 2)))
         Rows Removed by Filter: 11
         SubPlan 2
           ->  Result  (cost=87.73..87.74 rows=1 width=0) (actual time=1.957..1.957 rows=1 loops=8)
                 InitPlan 1 (returns $1)
                   ->  Limit  (cost=0.42..87.73 rows=1 width=2) (actual time=1.954..1.955 rows=1 loops=8)
                         ->  Index Only Scan Backward using permission_target_trashed on materialized_permissi
on_view  (cost=0.42..5413.51 rows=62 width=2) (actual time=1.952..1.952 rows=1 loops=8)
                               Index Cond: ((trashed IS NOT NULL) AND (target_uuid = (collections.owner_uuid):
:text))
                               Heap Fetches: 8
 Planning time: 0.357 ms
 Execution time: 15.806 ms
```

#11 - 12/04/2018 03:07 PM - Ward Vandewege

Changing the query to compare collections.uuid instead of collections.owner_uuid makes it use indexes and it is very fast.

The query is not equivalent, of course. But maybe we can add the corresponding indexes for the owner_uuid comparison?

```
arvados_production=# explain analyze SELECT  collections."uuid",collections.owner_uuid FROM "collections" WHER
E (NOT EXISTS(SELECT 1 FROM materialized_permission_view WHERE trashed = 1 and collections.uuid=target_uuid)
AND collections.is_trashed = false) ORDER BY collections.modified_at desc, collections.uuid LIMIT 8 OFFSET 0;
```

```
                                                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------
 Limit  (cost=0.84..7.56 rows=8 width=64) (actual time=0.037..0.111 rows=8 loops=1)
   ->  Nested Loop Anti Join  (cost=0.84..171327.50 rows=204054 width=64) (actual time=0.036..0.105 rows=8 loo
ps=1)
         ->  Index Scan using index_collections_on_modified_at_uuid on collections  (cost=0.42..68718.89 rows=
205487 width=64) (actual time=0.017..0.054 rows=8 loops=1)
               Filter: (NOT is_trashed)
               Rows Removed by Filter: 11
         ->  Index Only Scan using permission_target_trashed on materialized_permission_view  (cost=0.42..0.50
 rows=1 width=28) (actual time=0.006..0.006 rows=0 loops=8)
               Index Cond: ((trashed = 1) AND (target_uuid = (collections.uuid)::text))
               Heap Fetches: 0
 Planning time: 0.668 ms
 Execution time: 0.153 ms
(10 rows)
```

Incidentally, the results of both queries is actually identical in this particular case.

#12 - 12/04/2018 03:14 PM - Peter Amstutz

Tom Clegg wrote:

> Adjusting the "is trash?" subquery can make a big difference.
>
> [...]

I tried adding collections.uuid = collections.current_version_uuid back in:

```
EXPLAIN ANALYZE SELECT *
  FROM "collections"
  WHERE 1 IS DISTINCT FROM (SELECT MAX(trashed) FROM materialized_permission_view WHERE collections.owner_uuid
 = target_uuid)
    AND collections.is_trashed = false AND collections.uuid = collections.current_version_uuid
  ORDER BY collections.modified_at desc, collections.uuid
  LIMIT 8 OFFSET 0;

 Limit  (cost=94163.14..94163.16 rows=8 width=635) (actual time=411220.493..411220.497 rows=8 loops=1)
   ->  Sort  (cost=94163.14..94165.69 rows=1022 width=635) (actual time=411220.492..411220.492 rows=8 loops=1)
         Sort Key: collections.modified_at DESC, collections.uuid
         Sort Method: top-N heapsort  Memory: 32kB
         ->  Index Scan using index_collections_on_owner_uuid_and_name on collections  (cost=0.42..94142.70 ro
ws=1022 width=635) (actual time=15.070..410891.140 rows=205411 loops=1)
               Filter: (1 IS DISTINCT FROM (SubPlan 2))
               SubPlan 2
                 ->  Result  (cost=87.73..87.74 rows=1 width=0) (actual time=1.996..1.996 rows=1 loops=205411)
                       InitPlan 1 (returns $1)
                         ->  Limit  (cost=0.42..87.73 rows=1 width=2) (actual time=1.994..1.994 rows=1 loops=2
05411)
                               ->  Index Only Scan Backward using permission_target_trashed on materialized_pe
rmission_view  (cost=0.42..5413.51 rows=62 width=2) (actual time=1.993..1.993 rows=1 loops=205411)
                                     Index Cond: ((trashed IS NOT NULL) AND (target_uuid = (collections.owner_
uuid)::text))
                                     Heap Fetches: 205411
 Planning time: 0.363 ms
 Execution time: 411220.555 ms
```

#13 - 12/04/2018 03:28 PM - Peter Amstutz

```
explain analyze select *
FROM "collections"
WHERE collections.owner_uuid not in (SELECT target_uuid FROM materialized_permission_view WHERE trashed = 1)
AND collections.is_trashed = false
AND collections.uuid = collections.current_version_uuid
ORDER BY collections.modified_at desc, collections.uuid
LIMIT 8 OFFSET 0;

 Limit  (cost=2055.80..3141.49 rows=8 width=635) (actual time=0.074..0.116 rows=8 loops=1)
   ->  Index Scan using index_collections_on_modified_at_uuid on collections  (cost=2055.80..71811.94 rows=514
 width=635) (actual time=0.074..0.111 rows=8 loops=1)
         Filter: ((NOT is_trashed) AND (NOT (hashed SubPlan 1)) AND ((uuid)::text = (current_version_uuid)::te
xt))
```

```
         Rows Removed by Filter: 11
         SubPlan 1
           ->  Index Only Scan using permission_target_trashed on materialized_permission_view  (cost=0.42..20
51.76 rows=1447 width=28) (actual time=0.015..0.032 rows=12 loops=1)
                 Index Cond: (trashed = 1)
                 Heap Fetches: 12
 Planning time: 0.302 ms
 Execution time: 0.169 ms
```

#14 - 12/04/2018 03:44 PM - Peter Amstutz

14576-query-perf @ f8ea35a26789594253cff761dce87dd51ff9e89c

https://ci.curoverse.com/view/Developer/job/developer-run-tests/998/

#15 - 12/04/2018 03:52 PM - Tom Clegg

LGTM @ f8ea35a, thanks!

#16 - 12/04/2018 04:14 PM - Peter Amstutz

I think this makes a big difference for user queries as well.

explain analyze SELECT *
FROM "collections"
WHERE ((qr1hi-tpzed-327pde5hktairf9">EXISTS AND perm_level >= 1 AND trashed = 0 AND target_uuid = collections.uuid)
OR qr1hi-tpzed-327pde5hktairf9">EXISTS AND perm_level >= 1 AND trashed = 0 AND target_uuid = collections.owner_uuid AND
target_owner_uuid IS NOT NULL))
AND collections.is_trashed = false
AND collections.uuid = collections.current_version_uuid)
ORDER BY collections.modified_at desc, collections.uuid
LIMIT 8 OFFSET 0;

```
Limit  (cost=250556.39..250556.41 rows=8 width=635) (actual time=414.984..414.989 rows=8 loops=1)
   ->  Sort  (cost=250556.39..250558.31 rows=771 width=635) (actual time=414.982..414.983 rows=8 loops=1)
         Sort Key: collections.modified_at DESC, collections.uuid
         Sort Method: top-N heapsort  Memory: 31kB
         ->  Index Scan using index_collections_on_owner_uuid_and_name on collections  (cost=0.42..250540.97 r
ows=771 width=635) (actual time=1.984..389.860 rows=18391 loops=1)
               Filter: ((alternatives: SubPlan 1 or hashed SubPlan 2) OR (alternatives: SubPlan 3 or hashed Su
bPlan 4))
               Rows Removed by Filter: 187028
               SubPlan 1
                 ->  Index Scan using permission_target_trashed on materialized_permission_view  (cost=0.42..1
20.01 rows=1 width=0) (never executed)
                       Index Cond: ((trashed = 0) AND ((target_uuid)::text = (collections.uuid)::text))
                       Filter: ((perm_level >= 1) AND ((user_uuid)::text = 'qr1hi-tpzed-327pde5hktairf9
'::text))
               SubPlan 2
                 ->  Index Scan using permission_target_user_trashed_level on materialized_permission_view mat
erialized_permission_view_1  (cost=0.42..593.59 rows=352 width=28) (actual time=0.069..0.457 rows=212 loops=1)
                       Index Cond: (((user_uuid)::text = 'qr1hi-tpzed-327pde5hktairf9
'::text) AND (trashed = 0) AND (perm_level >= 1))
               SubPlan 3
                 ->  Index Scan using permission_target_trashed on materialized_permission_view materialized_p
ermission_view_2  (cost=0.42..120.01 rows=1 width=0) (never executed)
                       Index Cond: ((trashed = 0) AND ((target_uuid)::text = (collections.owner_uuid)::text))
                       Filter: ((target_owner_uuid IS NOT NULL) AND (perm_level >= 1) AND ((user_uuid)::text =
 'qr1hi-tpzed-327pde5hktairf9'::text))
               SubPlan 4
                 ->  Index Scan using permission_target_user_trashed_level on materialized_permission_view mat
erialized_permission_view_3  (cost=0.42..593.59 rows=283 width=28) (actual time=0.030..0.235 rows=190 loops=1)
                       Index Cond: (((user_uuid)::text = 'qr1hi-tpzed-327pde5hktairf9
'::text) AND (trashed = 0) AND (perm_level >= 1))
                       Filter: (target_owner_uuid IS NOT NULL)
                       Rows Removed by Filter: 22
 Planning time: 0.667 ms
 Execution time: 415.168 ms
```

```
 count
-------
 18391
```

SELECT *
FROM "collections"

```
WHERE ((collections.uuid in (SELECT target_uuid FROM materialized_permission_view WHERE user_uuid IN ('qr1hi-tpzed-327pde5hktairf9') AND perm_level >= 1 AND trashed = 0)
OR collections.owner_uuid in (SELECT target_uuid FROM materialized_permission_view WHERE user_uuid IN ('qr1hi-tpzed-327pde5hktairf9') AND perm_level >= 1 AND trashed = 0 AND target_owner_uuid IS NOT NULL)  )
AND collections.is_trashed = false
AND collections.uuid = collections.current_version_uuid)
ORDER BY collections.modified_at desc, collections.uuid
LIMIT 8 OFFSET 0;
```

```
 Limit  (cost=1189.19..1918.37 rows=8 width=635) (actual time=0.909..0.953 rows=8 loops=1)
   ->  Index Scan using index_collections_on_modified_at_uuid on collections  (cost=1189.19..71464.17 rows=771
 width=635) (actual time=0.907..0.946 rows=8 loops=1)
         Filter: ((NOT is_trashed) AND ((uuid)::text = (current_version_uuid)::text) AND ((hashed SubPlan 1) O
R (hashed SubPlan 2)))
         Rows Removed by Filter: 8
         SubPlan 1
           ->  Index Scan using permission_target_user_trashed_level on materialized_permission_view  (cost=0.
42..593.59 rows=352 width=28) (actual time=0.041..0.425 rows=212 loops=1)
                 Index Cond: (((user_uuid)::text = 'qr1hi-tpzed-327pde5hktairf9
'::text) AND (trashed = 0) AND (perm_level >= 1))
         SubPlan 2
           ->  Index Scan using permission_target_user_trashed_level on materialized_permission_view materiali
zed_permission_view_1  (cost=0.42..593.59 rows=283 width=28) (actual time=0.026..0.225 rows=190 loops=1)
                 Index Cond: (((user_uuid)::text = 'qr1hi-tpzed-327pde5hktairf9
'::text) AND (trashed = 0) AND (perm_level >= 1))
                 Filter: (target_owner_uuid IS NOT NULL)
                 Rows Removed by Filter: 22
 Planning time: 0.492 ms
 Execution time: 1.025 ms
(14 rows)
```

```
 count
-------
 18391
```

#17 - 12/04/2018 04:18 PM - Peter Amstutz

14576-query-perf @ 5090fa0891aa67f8373831a2c87ba69b65078b0d

Use the same optimization for user queries.

#18 - 12/04/2018 04:27 PM - Tom Clegg

Seems like the gist of this is that if we're going to use a subquery, it should not refer to any variables selected from the outer query. If the subquery is the same for every row, postgres only has to run it once.

#19 - 12/04/2018 04:29 PM - Tom Clegg

LGTM @ 5090fa0

#20 - 12/04/2018 08:42 PM - Ward Vandewege

- Status changed from New to Resolved

- Assigned To set to Peter Amstutz

- Target version set to 2018-12-12 Sprint

#21 - 02/21/2019 05:44 PM - Tom Morris

- Release set to 14