Напредни бази на податоци
Фаза 4 - Индекси и оптимизација на прашалници
Проект: LibraryDB

Андреј Станоев 231103
Тодор Темелков 231089
Благоја Ѓорлев 231062

View1 vw_active_memberships

1. Примарен филтер за погледот е според id на membership, дополнително ќе се користи id на корисник и email на корисник.
2. Примарен случај на употреба ќе е пребарување на активни членови.
3. Иницијално време на извршување 512 ms.

```
explain analyze
select * from vw_active_memberships where id = 5757215;
```

```
QUERY PLAN
Nested Loop  (cost=1.00..25.10 rows=1 width=229) (actual time=0.247..0.251 rows=1 loops=1)
  ->  Nested Loop  (cost=0.85..16.90 rows=1 width=83) (actual time=0.201..0.204 rows=1 loops=1)
        ->  Index Scan using membership_pkey on membership m  (cost=0.43..8.45 rows=1 width=40) (actual time=0.169..0.171 rows=1 loops=1)
              Index Cond: (id = 5757215)
              Filter: ((cancelled_at IS NULL) AND (expires_at > now()))
        ->  Index Scan using "User_pkey" on "User" u  (cost=0.43..8.45 rows=1 width=51) (actual time=0.027..0.027 rows=1 loops=1)
              Index Cond: (id = m.user_id)
  ->  Index Scan using membershipplan_pkey on membershipplan mp  (cost=0.15..8.17 rows=1 width=154) (actual time=0.044..0.044 rows=1 loops=1)
        Index Cond: (id = m.membership_plan_id)
Planning Time: 0.551 ms
Execution Time: 0.296 ms
11 rows
```

4. Ова е прифатливо време за извршување, нема потреба од додавање на индекси.

View2 vw_active_rentals

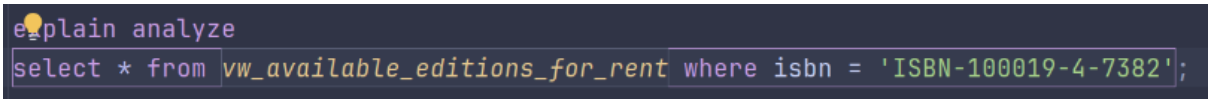
1. Примарен филтер за погледот е според id на rental, дополнително ќе се користи id на membership и име на издание.
2. Примарен случај на употреба ќе е пребарување на активно издадени книги.
3. Иницијално време на извршување 801 ms.

```
explain analyze
select * from vw_active_rentals where id = 6;
```

4. Прифатливо време, нема потреба од оптимизации.

View3 vw_available_editions_for_rent

1. Примарен филтер за погледот е според id на rental, дополнително ќе се користи id на membership и име на издание.
2. Примарен случај на употреба ќе е пребарување на книги кои што можат да се изнајмат.
3. Иницијално време на извршување 1s 424ms.



4. Овде премногу време трае делот за секвенцијално скенирање на копиите. Со тоа можеме да додадиме индекс на isbn во рамки на табелата за копиите.
5. Времето на извршување со додаден индекс изнесува 644ms.

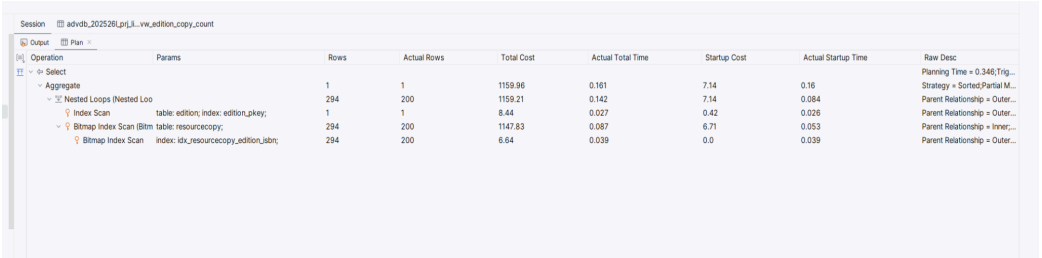
```
                  ->  Bitmap Heap Scan on resourcecopy rc  (cost=6.71..1147.83 rows=294 width=27) (actual time=0.097..0.147 rows=100 loops=1)
                        Recheck Cond: ((edition_isbn)::text = 'ISBN-100019-4-7382'::text)
                        Heap Blocks: exact=2
                        ->  Bitmap Index Scan on idx_resourcecopy_edition_isbn  (cost=0.00..6.64 rows=294 width=0) (actual time=0.055..0.055 rows=100 loops=1)
                              Index Cond: ((edition_isbn)::text = 'ISBN-100019-4-7382'::text)
                  ->  Hash  (cost=113.15..113.15 rows=4 width=16) (actual time=0.008..0.011 rows=0 loops=1)
                        Buckets: 1024  Batches: 1  Memory Usage: 8kB
                        ->  Nested Loop  (cost=5.85..113.15 rows=4 width=16) (actual time=0.008..0.009 rows=0 loops=1)
                              ->  Nested Loop  (cost=5.42..96.66 rows=4 width=24) (actual time=0.007..0.009 rows=0 loops=1)
                                    ->  Nested Loop  (cost=4.99..62.88 rows=4 width=24) (actual time=0.007..0.008 rows=0 loops=1)
                                          ->  Nested Loop  (cost=4.57..46.39 rows=4 width=43) (actual time=0.007..0.008 rows=0 loops=1)
                                                ->  Bitmap Heap Scan on rental r_1  (cost=4.13..12.58 rows=4 width=24) (actual time=0.007..0.007 rows=0 loops=…
                                                      Recheck Cond: (returned_at IS NULL)
                                                      ->  Bitmap Index Scan on idx_rental_active_resource_copy  (cost=0.00..4.13 rows=4 width=0) (actual time=…
                                                ->  Index Scan using resourcecopy_pkey on resourcecopy rc_1  (cost=0.43..8.45 rows=1 width=27) (never executed)
                                                      Index Cond: (id = r_1.resource_copy_id)
                                          ->  Index Only Scan using edition_pkey on edition e_1  (cost=0.42..4.12 rows=1 width=19) (never executed)
                                                Index Cond: (isbn = (rc_1.edition_isbn)::text)
                                                Heap Fetches: 0
                                    ->  Index Scan using membership_pkey on membership m  (cost=0.43..8.45 rows=1 width=16) (never executed)
                                          Index Cond: (id = r_1.membership_id)
                              ->  Index Only Scan using "User_pkey" on "User" u  (cost=0.43..4.12 rows=1 width=8) (never executed)
                                    Index Cond: (id = m.user_id)
                                    Heap Fetches: 0
        ->  Hash  (cost=12.58..12.58 rows=4 width=8) (actual time=0.007..0.008 rows=0 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 8kB
              ->  Bitmap Heap Scan on rental r  (cost=4.13..12.58 rows=4 width=8) (actual time=0.007..0.007 rows=0 loops=1)
                    Recheck Cond: (returned_at IS NULL)
                    ->  Bitmap Index Scan on idx_rental_active_resource_copy  (cost=0.00..4.13 rows=4 width=0) (actual time=0.002..0.003 rows=0 loops=1)
Planning Time: 3.342 ms
Execution Time: 0.477 ms
```

40 rows

View4 vw_edition_copy_count

1. Примарен филтер за погледот **vw_edition_copy_count** ќе биде според isbn на едицијата. Може да се користи и според името на едицијата.
2. Примарен случај на употреба на овој поглед е кога корисник или пак некој вработен би сакал да погледне од која едиција колку копии има.
3. Иницијално време на извршување на оваа квери е 547ms. Ова е прифатливо време, нема потреба од индекси.

```
1 ✓  select *
2    from vw_edition_copy_count as v
3    where v.isbn = 'ISBN-100002-5-74658';
```

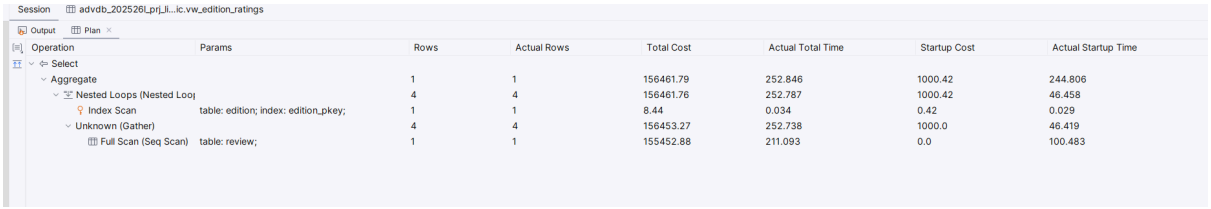


View5 vw_edition_ratings

1. Примарен филтер за погледот **vw_edition_ratings** ќе биде според isbn на едицијата. Може да се користи и според името на едицијата.

2. Примарен случај на употреба на овој поглед е кога корисник или пак некој вработен би сакал да виде просечната оцена за некоја едиција.
3. Иницијалното извршување на овој поглед е 1,6 s. Ова не е прифатливо време за апликацијата па поради тоа ќе пробаме да оптимизираме со индекси.

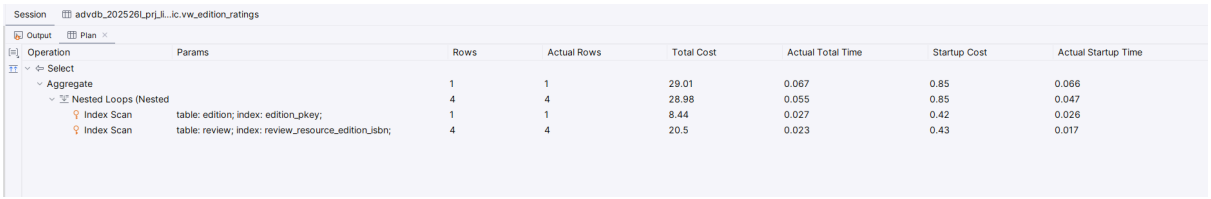
```
6  select *
7  from vw_edition_ratings as v
8  where v.isbn = 'ISBN-100000-8-85075';
9
```



Operation	Params	Rows	Actual Rows	Total Cost	Actual Total Time	Startup Cost	Actual Startup Time
Select							
Aggregate		1	1	156461.79	252.846	1000.42	244.806
Nested Loops (Nested Loo		4	4	156461.76	252.787	1000.42	46.458
Index Scan	table: edition; index: edition_pkey;	1	1	8.44	0.034	0.42	0.029
Unknown (Gather)		4	4	156453.27	252.738	1000.0	46.419
Full Scan (Seq Scan)	table: review;	1	1	155452.88	211.093	0.0	100.483

4. Бидејќи во табелата review имавме full scan, а во дефиницијата на погледот имаме join по review.resource_edition_isbn го создадив овој индекс и времето се намали до 600ms

```
create index review_resource_edition_isbn on review(resource_edition_isbn);
```

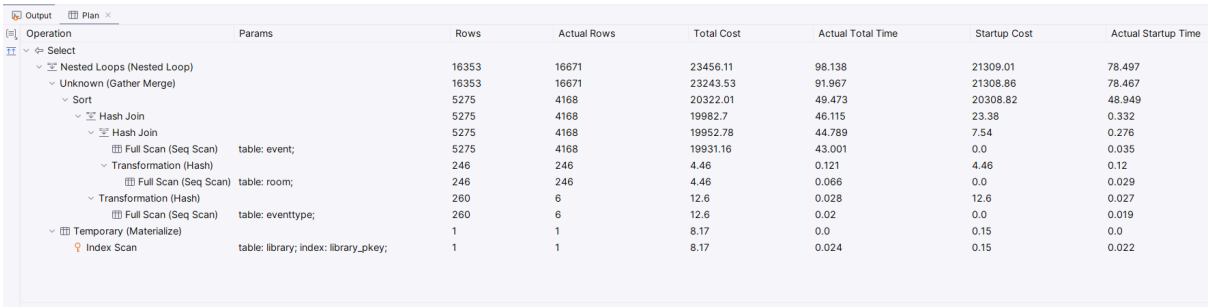


Operation	Params	Rows	Actual Rows	Total Cost	Actual Total Time	Startup Cost	Actual Startup Time
Select							
Aggregate		1	1	29.01	0.067	0.85	0.066
Nested Loops (Nested		4	4	28.98	0.055	0.85	0.047
Index Scan	table: edition; index: edition_pkey;	1	1	8.44	0.027	0.42	0.026
Index Scan	table: review; index: review_resource_edition_isbn;	4	4	20.5	0.023	0.43	0.017

View6 vw_upcoming_events

1. Примарен филтер за погледот **vw_upcoming_events** ќе биде по id на библиотека. А уште ќе се користи и според име на настан и тип на настан.
2. Примарен случај на употреба на овој поглед е кога корисник ќе сака да погледне кои се идни настани што ќе се случуваат во библиотеката.
3. Иницијално време на извршување на погледот е 1.8s. Ова не е прифатливо време за апликацијата па затоа ќе пробаме да оптимизираме со индекси.

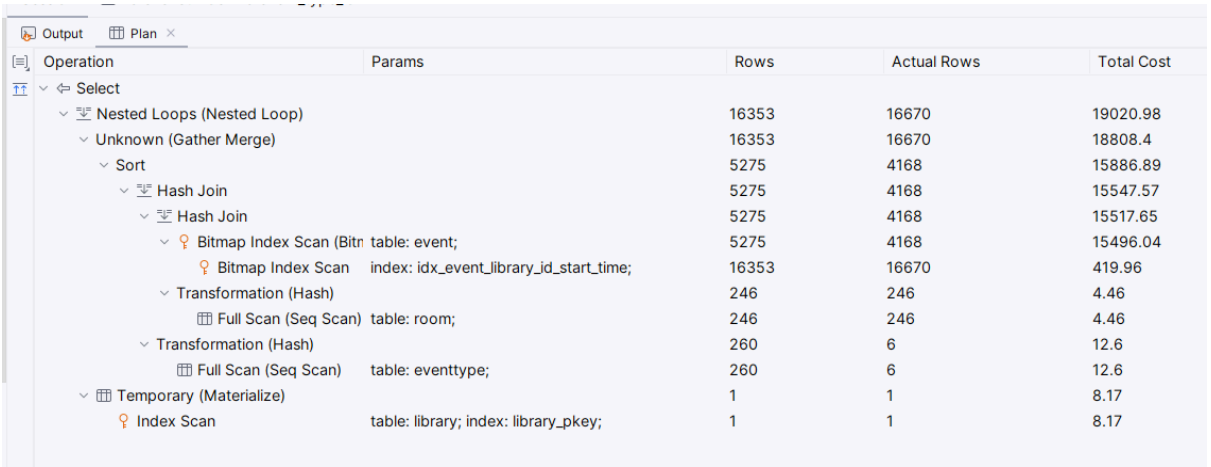
```sql
select *
from vw_future_events_for_lib as v
where v.library_id = 1;
```



Operation	Params	Rows	Actual Rows	Total Cost	Actual Total Time	Startup Cost	Actual Startup Time
Select							
Nested Loops (Nested Loop)		16353	16671	23456.11	98.138	21309.01	78.497
Unknown (Gather Merge)		16353	16671	23243.53	91.967	21308.86	78.467
Sort		5275	4168	20322.01	49.473	20308.82	48.949
Hash Join		5275	4168	19982.7	46.115	23.38	0.332
Hash Join		5275	4168	19952.78	44.789	7.54	0.276
Full Scan (Seq Scan)	table: event;	5275	4168	19931.16	43.001	0.0	0.035
Transformation (Hash)		246	246	4.46	0.121	4.46	0.12
Full Scan (Seq Scan)	table: room;	246	246	4.46	0.066	0.0	0.029
Transformation (Hash)		260	6	12.6	0.028	12.6	0.027
Full Scan (Seq Scan)	table: eventtype;	260	6	12.6	0.02	0.0	0.019
Temporary (Materialize)		1	1	8.17	0.0	0.15	0.0
Index Scan	table: library; index: library_pkey;	1	1	8.17	0.024	0.15	0.022

4. Бидејќи во погледот имаме дефенирато where >= now() а и спојуваме според ид на библиотека решив на создадам индекс каков што е одма подоле на сликата и сега погледот се извршува за 600ms.

```sql
create index idx_event_library_id_start_time on event(library_id, start_time);
```



Operation	Params	Rows	Actual Rows	Total Cost
Select				
Nested Loops (Nested Loop)		16353	16670	19020.98
Unknown (Gather Merge)		16353	16670	18808.4
Sort		5275	4168	15886.89
Hash Join		5275	4168	15547.57
Hash Join		5275	4168	15517.65
Bitmap Index Scan (Bitn	table: event;	5275	4168	15496.04
Bitmap Index Scan	index: idx_event_library_id_start_time;	16353	16670	419.96
Transformation (Hash)		246	246	4.46
Full Scan (Seq Scan)	table: room;	246	246	4.46
Transformation (Hash)		260	6	12.6
Full Scan (Seq Scan)	table: eventtype;	260	6	12.6
Temporary (Materialize)		1	1	8.17
Index Scan	table: library; index: library_pkey;	1	1	8.17

5. Не додавам идекси за event_room_id и event_type_id бидејќи тие две табели имаат малце редови.

View7 vw_overdue_rentals

1.Како примарен филтер би го земале id од табелата resource исто така може да се пребарува и според email.
2.Примарен случај на употреба би било корисникот да добие информација која книга до кога треба да ја врати.
3.Инцијалното време на извршување е 485ms што е прифатливо односно нема потреба за додавање на индекси.

```
select * from vw_overdue_rentals
where id = 535211;
```

Output | Query Plan | advdb_202526l_prj_li...ic.vw_overdue_rentals

Operation	Params	Rows	Actual Rows	Total Cost	Actual Total Time	Startup Cost	Actual Startup Time
Select							
Nested Loops (Nested Lc		1	1	40.68	0.289	2.15	0.285
Nested Loops (Nestec		1	1	33.02	0.235	1.72	0.231
Nested Loops (Nes		1	1	24.58	0.169	1.29	0.166
Nested Loops (I		1	1	16.91	0.15	0.87	0.147
Index Scan	table: rental; index: rental_pkey;	1	1	8.46	0.076	0.43	0.075
Index Scan	table: resourcecopy; index: resourcecopy_pkey;	1	1	8.45	0.069	0.43	0.069
Index Scan	table: edition; index: edition_pkey;	1	1	7.67	0.017	0.42	0.017
Index Scan	table: membership; index: membership_pkey;	1	1	8.45	0.063	0.43	0.063
Index Scan	table: User; index: User_pkey;	1	1	7.66	0.052	0.43	0.052

Consoles > advdb_202526l_prj_libdb > console_1 [advdb_202526l_prj_libdb]

View8 vw_sorted_by_rentals_rented

1.Како примарен филтер ке го искористиме ISBN од табелата edition дополнително можеме да пребраме преку name од табелата edition.

2.Примарен случај на употреба би бил да добие прегледност корисникот во тоа кој се најпопуларни изнајмени книги.

3.Иницијално време на извршување е 1.746s.

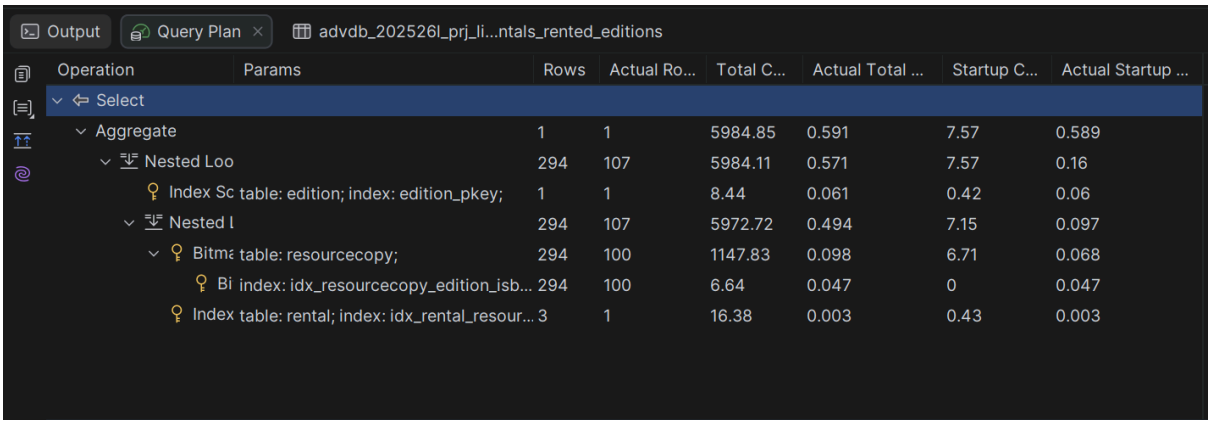
```
select * from vw_sorted_by_rentals_rented_editions
where isbn = 'ISBN-155380-8-87703';
```

4.Овде најголем проблем ни претставува секвенцијалниот скен во табелата rental затоа додаваме индекс во табелата rental на колоната resource_copy_id.

Output | Query Plan | advdb_202526l_prj_li...ntals_rented_editions

Operation	Params	Rows	Actual Ro...	Total C...	Actual Total T...	Startup C...	Actual Startup ...
Select							
Aggregate		1	1	156606.89	15189.246	2151.93	15180.353
Nested Loops (N		294	107	156606.15	15189.176	2151.93	15078.102
Index Scan	table: edition; index: edition_p..	1	1	8.44	0.111	0.42	0.106
Unknown (Gathe		294	107	156594.76	15189.038	2151.51	15077.986
Hash Join		122	36	155565.36	15147.283	1151.51	14618.499
Full Sca	table: rental;	4166...	3333333	143476.42	14782.638	0	100.407
Transform		294	100	1147.83	12.088	1147.83	12.086
Bitm	table: resourcecopy;	294	100	1147.83	12.038	6.71	12.01

5.Времето на извршување по додадениот индекс изнесува 391ms.



Operation	Params	Rows	Actual Ro...	Total C...	Actual Total ...	Startup C...	Actual Startup ...
Select							
Aggregate		1	1	5984.85	0.591	7.57	0.589
Nested Loo		294	107	5984.11	0.571	7.57	0.16
Index Sc	table: edition; index: edition_pkey;	1	1	8.44	0.061	0.42	0.06
Nested l		294	107	5972.72	0.494	7.15	0.097
Bitma	table: resourcecopy;	294	100	1147.83	0.098	6.71	0.068
Bi	index: idx_resourcecopy_edition_isb...	294	100	6.64	0.047	0	0.047
Index	table: rental; index: idx_rental_resour...	3	1	16.38	0.003	0.43	0.003

View9 vw_user_activiy

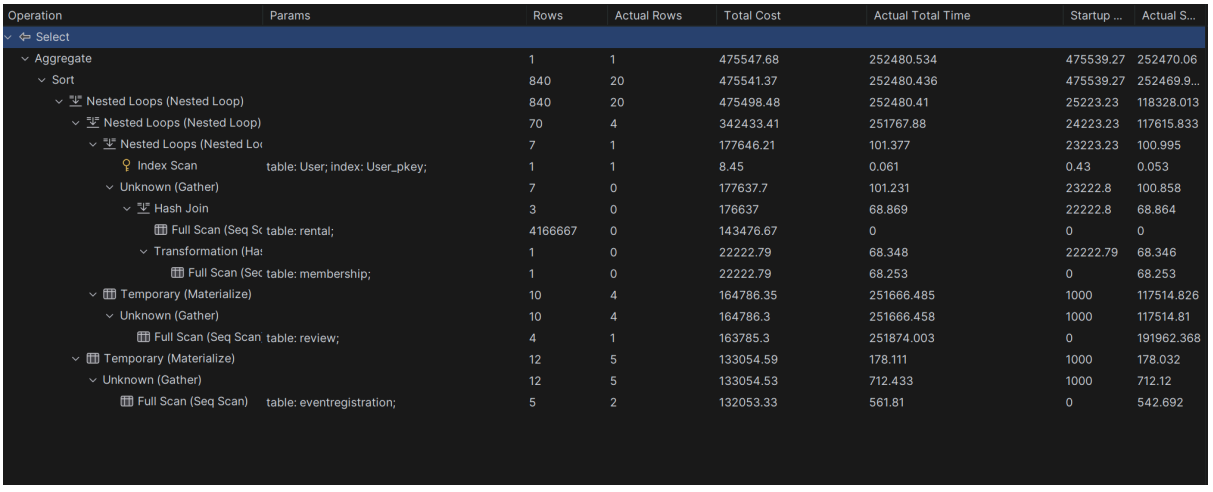
1.Како примарен филтер ќе го искористиме user_id дополнително можеме да пребаруваме според email.

2.Примарен случај на употреба би било корисникот да добие преглед во своите податоци во библиотекате како колку пати изнајмил колку пати оставил оцена на книга и на колку настани учествувал.

3.Иницијалното време на извршување претставува 4.899s.

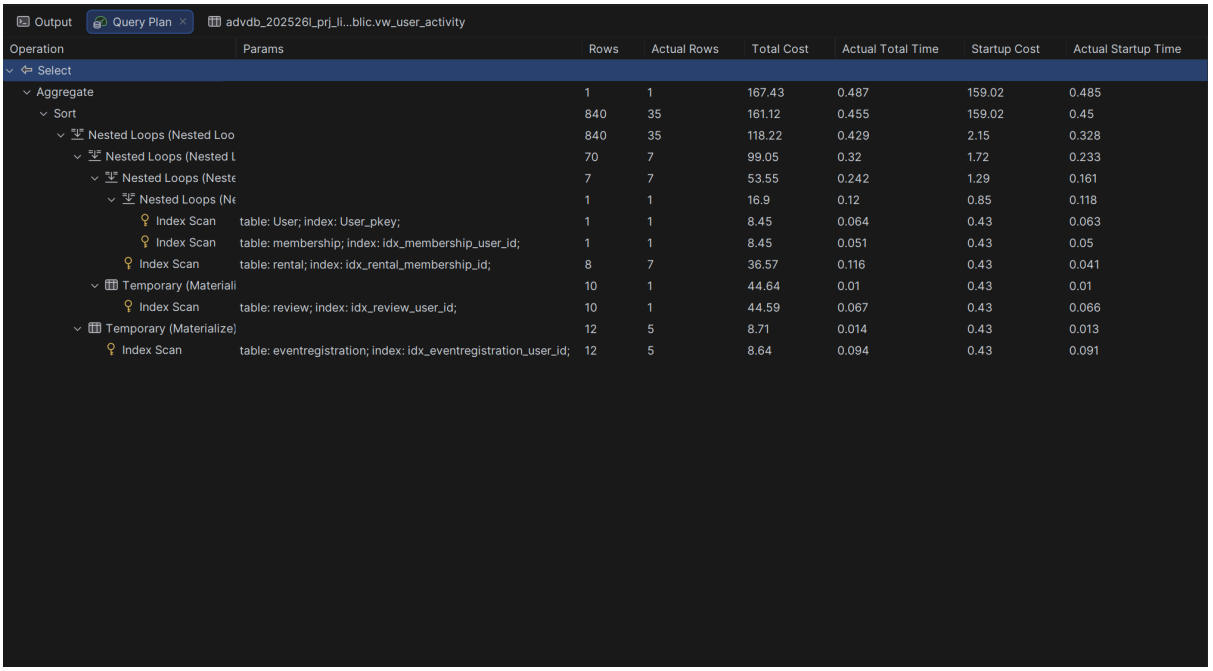
```
select * from vw_user_activity
where user_id = 1
```

4.Пребарувањето трае премногу време поради тоа што имаме 4 различни секвенцијални скенирања на табелите.
rental,membership,review,eventregistration за да го решиме овој проблем можеме да додадеме индекс на user_id во табелите membership,review и eventregistration, а во табелата rental да додадеме индекс на membership_id.



Operation	Params	Rows	Actual Rows	Total Cost	Actual Total Time	Startup ...	Actual S...
Select							
Aggregate		1	1	475547.68	252480.534	475539.27	252470.06
Sort		840	20	475541.37	252480.436	475539.27	252469.9...
Nested Loops (Nested Loop)		840	20	475498.48	252480.41	25223.23	118328.013
Nested Loops (Nested Loop)		70	4	342433.41	251767.88	24223.23	117615.833
Nested Loops (Nested Loo		7	1	177646.21	101.377	23223.23	100.995
Index Scan	table: User; index: User_pkey;	1	1	8.45	0.061	0.43	0.053
Unknown (Gather)		7	0	177637.7	101.231	23222.8	100.858
Hash Join		3	0	176637	68.869	22222.8	68.864
Full Scan (Seq Sc	table: rental;	4166667	0	143476.67	0	0	0
Transformation (Ha		1	0	22222.79	68.348	22222.79	68.346
Full Scan (Sec	table: membership;	1	0	22222.79	68.253	0	68.253
Temporary (Materialize)		10	4	164786.35	251666.485	1000	117514.826
Unknown (Gather)		10	4	164786.3	251666.458	1000	117514.81
Full Scan (Seq Scan	table: review;	4	1	163785.3	251874.003	0	191962.368
Temporary (Materialize)		12	5	133054.59	178.111	1000	178.032
Unknown (Gather)		12	5	133054.53	712.433	1000	712.12
Full Scan (Seq Scan)	table: eventregistration;	5	2	132053.33	561.81	0	542.692

5.Со додавање на овие индекси времето паѓа на прифатливи 350ms.



Operation	Params	Rows	Actual Rows	Total Cost	Actual Total Time	Startup Cost	Actual Startup Time
Select							
Aggregate		1	1	167.43	0.487	159.02	0.485
Sort		840	35	161.12	0.455	159.02	0.45
Nested Loops (Nested Loo		840	35	118.22	0.429	2.15	0.328
Nested Loops (Nested L		70	7	99.05	0.32	1.72	0.233
Nested Loops (Neste		7	7	53.55	0.242	1.29	0.161
Nested Loops (Ne		1	1	16.9	0.12	0.85	0.118
Index Scan	table: User; index: User_pkey;	1	1	8.45	0.064	0.43	0.063
Index Scan	table: membership; index: idx_membership_user_id;	1	1	8.45	0.051	0.43	0.05
Index Scan	table: rental; index: idx_rental_membership_id;	8	7	36.57	0.116	0.43	0.041
Temporary (Materiali		10	1	44.64	0.01	0.43	0.01
Index Scan	table: review; index: idx_review_user_id;	10	1	44.59	0.067	0.43	0.066
Temporary (Materialize)		12	5	8.71	0.014	0.43	0.013
Index Scan	table: eventregistration; index: idx_eventregistration_user_id;	12	5	8.64	0.094	0.43	0.091