

ИНДЕКСИ И ОПТИМИЗАЦИЈА НА ПРАШАЛНИЦИ

Елена Спасовска 231141
Марија Сергиевска 231048
Кристијан Тримчески 231132

View 1: Worker Global Rating

1. Примарен филтер за материјализираниот поглед ќе биде според `worker_id`
2. Погледот ќе се користи за прикажување на просечниот рејтинг и бројот на `reviews` за одреден `worker`. Со користење на материјализиран поглед се избегнува повторно пресметување на агрегатните вредности при секоја отварање на `worker profile`.
3. Времето на извршување на прашалникот е 18ms, што е прифатливо време за апликацијата.
4. Во планот за извршување се користи `Parallel Seq Scan` врз `worker_rating_summary`. Иако се прави `sequential scan`, оптимизаторот проценил дека тоа е поефикасно решение за моменталната големина на податоците.
5. Бројот на записи е релативно мал, а времето на извршување е многу ниско. Нема потреба од дополнително индексирање бидејќи времето на извршување е задоволително, материјализираниот поглед веќе ги чува пресметаните агрегатни функции, дополнителен индекс не би донел значително подобрување на перформансите.

```
EXPLAIN ANALYZE
SELECT *
FROM worker_rating_summary
WHERE worker_id = 95
```

AZ QUERY PLAN
Gather (cost=1000.00..4246.34 rows=1 width=22) (actual time=2.609..18.424 rows=1 loops=1)
Workers Planned: 1
Workers Launched: 1
-> Parallel Seq Scan on worker_rating_summary (cost=0.00..3246.24 rows=1 width=22) (actual time=4.841..10.710 rows=0 loops=2)
Filter: (worker_id = 95)
Rows Removed by Filter: 125000
Planning Time: 0.089 ms
Execution Time: 18.462 ms

View 2: Worker Category Rating

1. Примарен филтер за `materializer view worker_rating_summary` ќе биде според `worker_id`, а дополнително ќе може да се пребарува и според `category_id`
2. Погледот ќе се користи за прикажување на просечниот рејтинг и бројот на `reviews` за одреден `worker` по категории, како и бројот на `reviews` за секоја категорија. Со тоа се избегнува повторно пресметување на оценките при секоја отварање на `worker профилот`. Ова е важно поради тоа што овие информации се користат при `matching` процесот, како и при прикажување на `badge` и `rating` информациите во системот.
3. Времето на извршување на прашалникот изнесува 5.757ms, што е одлично и прифатливо време на извршување.
4. Нема потреба од дополнително индексирање бидејќи времето на извршување е многу мало, материјализираниот поглед веќе ги содржи предходно пресметаните агрегатни вредности. Дополнителен индекс не би продонел за значително подобрување на брзината на извршување.

```
EXPLAIN ANALYZE
SELECT *
FROM worker_category_rating_summary
WHERE worker_id = 95
```

AZ QUERY PLAN
Seq Scan on worker_category_rating_summary (cost=0.00..1304.53 rows=1 width=22) (actual time=0.040..5.737 rows=2 loops=1)
Filter: (worker_id = 95)
Rows Removed by Filter: 68520
Planning Time: 0.188 ms
Execution Time: 5.757 ms

View 3: Client Global Rating

1. Примарен филтер за материјализираниот поглед ќе биде според `client_id`
2. Погледот ќе се користи за прикажување на просечниот рејтинг и вкупниот број на `reviews` за одреден `client`. Овде се важни перформансите, бидејќи овие информации често се прикажуваат во системот.
3. Времето на извршување на прашалникот изнесува 6.918ms, што е одлично и прифатливо време
4. Во планот за извршување се користи `Seq Scan` врз `client_rating_summary`.
5. Нема потреба од дополнително индексирање бидејќи времето на извршување е многу мало, материјализираниот поглед ги содржи предходно пресметаните агрегатни вредности, дополнителен индекс не би донел значително подобрување на перформансите.

```
EXPLAIN ANALYZE
SELECT *
FROM client_rating_summary
WHERE client_id=1
```

AZ QUERY PLAN
Seq Scan on client_rating_summary (cost=0.00..1890.00 rows=1 width=22) (actual time=1.263..6.898 rows=1 loops=1)
Filter: (client_id = 1)
Rows Removed by Filter: 99999
Planning Time: 0.069 ms
Execution Time: 6.918 ms

View 4: Worker Profile View

1. Примарен филтер за погледот ќе биде според `worker_id`
2. Погледот ќе се користи за прикажување на основни информации за `worker`, неговата локација и `rating summary` информации.
3. Времето на извршување е 20.535ms, што е прифатливо и брзо време за апликацијата
4. Во планот на извршување се користи `Index Scan` врз `worker`, `useraccount`, `location`. Ова овозможува ефикасно пребарување по вредностите на примарниот клуч.
5. Се користи `Parallel Seq Scan` на `worker_rating_summary`. Иако имаме `Parallel Seq Scan`, времето на извршување останува исто.
6. Материјализираниот поглед `worker_rating_summary` веќе ги содржи претходно пресметаните агрегатни вредности, што дополнително ги подобрува перформансите.
7. Нема потреба од оптимизација бидејќи времето на извршување е прифатливо

```
EXPLAIN ANALYZE
SELECT *
FROM worker_profile_view
WHERE worker_id = 95
```

AZ QUERY PLAN
Nested Loop Left Join (cost=1001.13..4271.53 rows=1 width=117) (actual time=3.170..20.445 rows=1 loops=1)
-> Nested Loop (cost=1.13..25.18 rows=1 width=103) (actual time=0.032..0.037 rows=1 loops=1)
-> Nested Loop (cost=0.84..16.88 rows=1 width=83) (actual time=0.024..0.028 rows=1 loops=1)
-> Index Scan using worker_pkey on worker w (cost=0.42..8.44 rows=1 width=23) (actual time=0.011..0.014 rows=1 loops=1)
Index Cond: (id = 95)
-> Index Scan using useraccount_pkey on useraccount ua (cost=0.42..8.44 rows=1 width=64) (actual time=0.008..0.008 rows=1 loops=1)
Index Cond: (id = w.user_id)
-> Index Scan using location_pkey on location l (cost=0.29..8.30 rows=1 width=28) (actual time=0.007..0.007 rows=1 loops=1)
Index Cond: (id = w.location_id)
-> Gather (cost=1000.00..4246.34 rows=1 width=18) (actual time=3.136..20.404 rows=1 loops=1)
Workers Planned: 1
Workers Launched: 1
-> Parallel Seq Scan on worker_rating_summary wrs (cost=0.00..3246.24 rows=1 width=18) (actual time=5.743..12.276 rows=0 loops=2)
Filter: (worker_id = 95)
Rows Removed by Filter: 125000
Planning Time: 0.680 ms
Execution Time: 20.535 ms

View 5: Available Task Request

1. Примарен филтер за погледот ќе биде според worker_id, а дополнително се врши филтрирање според category, location, distance и status на task request.
2. Погледот ќе се користи за прикажување на сите достапни task requests за одреден worker. Оваа функционалност е важна за системот, бидејќи workers постојано пребаруваат нови задачи.
3. Времето на извршување на прашалникот изнесува 69980ms (~70 секунди), што не е прифатливо време за апликацијата.
4. Во планот на извршување најголемиот проблем претставува Parallel Seq Scan врз TaskRequest и скапите географски пресметки со ACOS, COS и SIN
5. Мора да се скенира огромен број TaskRequest записи за да ги пронајде само OPEN task requests, matching категории и worker кои се во дозволен радиус.
6. За оптимизација е додаден композитен индекс:

```
CREATE INDEX idx_taskrequest_status_category  
ON TaskRequest(status, category_id);
```

7. По индексирањето започна да се користи индексот и имаме Index Scan, наместо full sequential scan врз TaskRequest и времето на извршување на прашалникот изнесува 59.890ms што е прифатливо и брзо време за апликацијата

пред индексирање:

```
EXPLAIN ANALYZE  
SELECT *  
FROM available_task_requests_view  
WHERE worker_id = 95
```

Az QUERY PLAN	
Hash Cond: (tr.category_id = wc.category_id)	
-> Parallel Seq Scan on taskrequest tr (cost=0.00..122705.69 rows=52684 width=95) (actual time=16.916..67428.971 rows=29924 loops=1)	
Filter: ((deleted_at IS NULL) AND ((status)::text = 'OPEN'::text))	
Rows Removed by Filter: 973576	
-> Hash (cost=4.48..4.48 rows=3 width=8) (actual time=0.096..0.097 rows=3 loops=5)	
Buckets: 1024 Batches: 1 Memory Usage: 9kB	
-> Index Only Scan using unq_worker_category on workercategory wc (cost=0.42..4.48 rows=3 width=8) (actual time=0.084..0.085 rows=3 loops=5)	
Index Cond: (worker_id = 95)	
Heap Fetches: 0	
-> Hash (cost=6.13..6.13 rows=213 width=26) (actual time=0.143..0.148 rows=213 loops=5)	
Buckets: 1024 Batches: 1 Memory Usage: 21kB	
-> Seq Scan on category cat (cost=0.00..6.13 rows=213 width=26) (actual time=0.050..0.098 rows=213 loops=5)	
-> Index Scan using location_pkey on location l (cost=0.29..0.30 rows=1 width=28) (actual time=0.318..0.318 rows=1 loops=1888)	
Index Cond: (id = tr.location_id)	
-> Index Scan using client_pkey on client c (cost=0.29..0.32 rows=1 width=8) (actual time=0.915..0.915 rows=1 loops=671)	
Index Cond: (id = tr.client_id)	
-> Index Scan using useraccount_pkey on useraccount ua (cost=0.42..0.66 rows=1 width=19) (actual time=16.381..16.381 rows=1 loops=671)	
Index Cond: (id = c.user_id)	
Planning Time: 5.608 ms	
JIT:	
Functions: 150	
Options: Inlining false, Optimization false, Expressions true, Deforming true	
Timing: Generation 12.172 ms (Deform 5.882 ms), Inlining 0.000 ms, Optimization 5.128 ms, Emission 113.443 ms, Total 130.743 ms	
Execution Time: 69980.539 ms	

после индексирање:

Az QUERY PLAN	
Index Cond: (id = w.location_id)	
-> Hash Join (cost=4.51..11.22 rows=3 width=34) (actual time=0.145..0.214 rows=3 loops=1)	
Hash Cond: (cat.id = wc.category_id)	
-> Seq Scan on category cat (cost=0.00..6.13 rows=213 width=26) (actual time=0.035..0.071 rows=213 loops=1)	
-> Hash (cost=4.48..4.48 rows=3 width=8) (actual time=0.069..0.070 rows=3 loops=1)	
Buckets: 1024 Batches: 1 Memory Usage: 9kB	
-> Index Only Scan using unq_worker_category on workercategory wc (cost=0.42..4.48 rows=3 width=8) (actual time=0.057..0.060 rows=3 loops=1)	
Index Cond: (worker_id = 95)	
Heap Fetches: 0	
-> Index Scan using idx_taskrequest_status_category on taskrequest tr (cost=0.43..1166.08 rows=1092 width=95) (actual time=0.078..8.978 rows=629 loops=3)	
Index Cond: (((status)::text = 'OPEN'::text) AND (category_id = cat.id))	
Filter: (deleted_at IS NULL)	
-> Hash (cost=182.00..182.00 rows=10400 width=28) (actual time=5.393..5.394 rows=10400 loops=1)	
Buckets: 16384 Batches: 1 Memory Usage: 780kB	
-> Seq Scan on location l (cost=0.00..182.00 rows=10400 width=28) (actual time=0.037..2.134 rows=10400 loops=1)	
-> Index Scan using client_pkey on client c (cost=0.29..0.32 rows=1 width=8) (actual time=0.005..0.005 rows=1 loops=671)	
Index Cond: (id = tr.client_id)	
-> Index Scan using useraccount_pkey on useraccount ua (cost=0.42..0.66 rows=1 width=19) (actual time=0.013..0.013 rows=1 loops=671)	
Index Cond: (id = c.user_id)	
Planning Time: 10.613 ms	
Execution Time: 59.890 ms	

View 6: Client Profile View

1. Примарен филтер за погледот ќе биде според `client_id`
2. Погледот ќе се користи за прикажување на основните информации за `client` и неговиот `rating summary`
3. Времето на извршување изнесува 18.485ms, што е прифатливо и брзо време за апликацијата
4. Во планот за извршување се користи `Index Scan` врз `client` и `useraccount`, што овозможува пребарување по вредноста на примарниот клуч
5. Се користи `Seq Scan` на `client_rating_summary`, но тоа не претставува проблем.
6. Материјализираниот поглед `client_rating_summary` веќе ги содржи претходно пресметаните агрегатни вредности, што дополнително ги подобрува перформансите

```
EXPLAIN ANALYZE
SELECT *
FROM client_profile_view
WHERE client_id=1
```

AZ QUERY PLAN	
Hash Right Join	(cost=16.76..2031.77 rows=1 width=108) (actual time=3.433..18.362 rows=1 loops=1)
Hash Cond:	(crs.user_id = ua.id)
-> Seq Scan on client_rating_summary crs	(cost=0.00..1640.00 rows=100000 width=18) (actual time=0.030..8.099 rows=100000 loops=1)
-> Hash	(cost=16.75..16.75 rows=1 width=68) (actual time=0.095..0.097 rows=1 loops=1)
Buckets:	1024 Batches: 1 Memory Usage: 9kB
-> Nested Loop	(cost=0.71..16.75 rows=1 width=68) (actual time=0.085..0.088 rows=1 loops=1)
-> Index Scan using client_pkey on client c	(cost=0.29..8.31 rows=1 width=8) (actual time=0.041..0.042 rows=1 loops=1)
Index Cond:	(id = 1)
-> Index Scan using useraccount_pkey on useraccount ua	(cost=0.42..8.44 rows=1 width=64) (actual time=0.039..0.039 rows=1 loops=1)
Index Cond:	(id = c.user_id)
Planning Time:	2.029 ms
Execution Time:	18.485 ms

View 7: Matched Workers View

1. Примарен филтер за погледот ќе биде според `task_request_id`, а дополнително се врши филтрирање и според статус на `task request (OPEN)`, `category matching`, `work mode`, `active badges`, `favourite статус` и `географска оддалеченост`.
2. Времето на извршување изнесува 424.172ms, што претставува прифатливо време за `complex matching` функционалност
3. Query-то дополнително користи `complex Nested Loop`, `Hash Join`, `Merge Join` и `Parallel Hash Join`, како и пресметки за оддалеченост, и повеќе `Join` операции, што дополнително ја зголемува сложеноста на планот за извршување.
4. Најголем трошок создаваат `Parallel Seq Scan on worker badge`, `Parallel Seq Scan on worker`, `Seq Scan on worker_category_rating_summary`, но и покрај ова и комплексното query, времето на извршување е прифатливо за `matching` функционалност

```
EXPLAIN ANALYZE
SELECT *
FROM matched_workers_view
WHERE task_request_id = 118
```

AZ QUERY PLAN	
-> Seq Scan on category cat	(cost=0.00..6.13 rows=213 width=26) (actual time=0.033..0.067 rows=213 loops=3)
-> Hash	(cost=2.90..2.90 rows=90 width=48) (actual time=0.069..0.069 rows=90 loops=1)
Buckets:	1024 Batches: 1 Memory Usage: 16kB
-> Seq Scan on badge b	(cost=0.00..2.90 rows=90 width=48) (actual time=0.013..0.033 rows=90 loops=1)
-> Hash	(cost=15.38..15.38 rows=938 width=12) (actual time=0.303..0.304 rows=938 loops=1)
Buckets:	1024 Batches: 1 Memory Usage: 49kB
-> Seq Scan on favourite f	(cost=0.00..15.38 rows=938 width=12) (actual time=0.012..0.113 rows=938 loops=1)
Planning Time:	19.064 ms
Execution Time:	424.172 ms

View 8: Task Request Offers View

1. Примарен филтер за погледот ќе биде спроед `task_request_id`
2. Погледот ќе се користи за прикажување на сите понуди (offers) за одреден task request, заедно со информациите за worker, rating summary, badges, favourite статус, географска оддалеченост и рангирање според цена, rating и оддалеченост.
3. Иницијалното време на извршување на query-то изнесува 272714.894ms (~4.5 минути), што е неприфатливо време за апликацијата.
4. Најголем проблем во планот на извршување претставува Parallel Seq Scan on offer o.
5. Дополнителен трошок создава и Seq Scan on worker_rating_summary, кој се извршува повеќе пати (loops=16)
6. Query-то дополнително користи комплексни Nested Loop операции, distance calculation, агрегатни функции и повеќе Join операции
7. За оптимизација е додаден композитен индекс:
**CREATE INDEX idx_offer_status_taskrequest
ON Offer(offer_status, task_request_id);**
8. По индексирањето започна да се користи индексот и имаме Index Scan, наместо Parallel Seq Scan врз Offer и времето на извршување на прашалникот изнесува 261.001ms што е прифатливо и брзо време за апликацијата

пред индексирање:

```
EXPLAIN ANALYZE  
SELECT *  
FROM task_request_offers_view  
WHERE task_request_id = 5000001
```

1 x

N ANALYZE SELECT * FROM task_request_offers_view

AZ QUERY PLAN

Join Filter: (o.worker_id = wrs.worker_id)
Rows Removed by Join Filter: 3999984
-> Gather (cost=1000.00..582364.21 rows=1 width=31) (actual time=254090.724..254090.944 rows=16 loops=1)
Workers Planned: 4
Workers Launched: 4
-> Parallel Seq Scan on offer o (cost=0.00..581364.11 rows=1 width=31) (actual time=152538.245..254052.395 rows=3 loops=1)
Filter: ((deleted_at IS NULL) AND (task_request_id = 5000001) AND ((offer_status)::text = 'PENDING'::text))
Rows Removed by Filter: 3179043
-> Seq Scan on worker_rating_summary wrs (cost=0.00..3908.00 rows=250000 width=18) (actual time=0.047..51.940 rows=16 loops=16)
-> Index Scan using worker_pkey on worker w (cost=0.42..1.94 rows=1 width=19) (actual time=1.157..1.157 rows=1 loops=16)

Execution Time: 272714.894 ms

после индексирање:

-> Parallel Seq Scan on worker_rating_summary wrs (cost=0.00..2878.59 rows=147059 width=18) (actual time=0.047..51.940 rows=16 loops=16)
-> Hash (cost=8.09..8.09 rows=1 width=31) (actual time=0.122..0.126 rows=16 loops=2)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Index Scan using idx_offer_status_taskrequest on offer o (cost=0.56..8.09 rows=1 width=31) (actual time=1.157..1.157 rows=1 loops=1)
Index Cond: (((offer_status)::text = 'PENDING'::text) AND (task_request_id = 5000001))
Filter: (deleted_at IS NULL)
-> Index Scan using worker_pkey on worker w (cost=0.42..1.94 rows=1 width=19) (actual time=1.157..1.157 rows=1 loops=16)
Index Cond: (id = wrs.worker_id)
-> Index Scan using useraccount_pkey on useraccount ua (cost=0.42..1.97 rows=1 width=19) (actual time=1.157..1.157 rows=1 loops=16)
Index Cond: (id = w.user_id)
-> Index Scan using location_pkey on location wl (cost=0.29..1.67 rows=1 width=20) (actual time=0.005..0.005 rows=1 loops=16)
Index Cond: (id = w.location_id)
-> Index Scan using taskrequest_pkey on taskrequest tr (cost=0.43..8.45 rows=1 width=16) (actual time=1.157..1.157 rows=1 loops=16)
Index Cond: (id = 5000001)
-> Index Scan using location_pkey on location trl (cost=0.29..8.30 rows=1 width=20) (actual time=0.005..0.005 rows=1 loops=16)
Index Cond: (id = tr.location_id)
-> Index Scan using unq_worker_badge on workerbadge wb (cost=0.42..9991.32 rows=3 width=8) (actual time=0.003..0.003 rows=1 loops=16)
Index Cond: (worker_id = w.id)
Filter: is_active
Rows Removed by Filter: 0
-> Index Scan using badge_pkey on badge b (cost=0.14..0.16 rows=1 width=40) (actual time=0.004..0.005 rows=1 loops=16)
Index Cond: (id = wb.badge_id)
-> Index Scan using unq_worker_client on favourite f (cost=0.28..1.27 rows=2 width=12) (actual time=0.003..0.003 rows=1 loops=16)
Index Cond: (worker_id = w.id)

Planning Time: 10.374 ms

Execution Time: 261.001 ms

View 9: Active Task View

1. Примарен филтер за погледот ќе биде според `worker_id`, а дополнително се користи и филтер според `status`
2. Погледот ќе се користи за прикажување на сите активни задачи за одреден `worker`, заедно со информации за `client`, `category`, `location` и цената на `task`-от.
3. Иницијалното време на извршување на `query`-то изнесува `18971.880ms` (~19 секунди), што е неприфатливо време за апликацијата.
4. Најголем проблем во планот на извршување претставуваат `Parallel Seq Scan on offer` и `Parallel Seq Scan on task`, при што се обработуваат огромен број записи за да ги пронајде `offers` за конкретен `worker` и `tasks` со статус `ACTIVE`.
5. `Query`-то дополнително користи `Hash Join`, повеќе `Nested Loop` операции и повеќе `JOIN` операции, што дополнително ја зголемува сложеноста на `execution plan`-от.
6. За оптимизација е додаден индексот:
CREATE INDEX idx_offer_worker ON Offer(worker_id);
7. Со индексирање започна користење на `Bitmap Index Scan on idx_offer_worker`, времето на извршување на `query`-то се намали на `193.597ms` и ова претставува огромно подобрување и прифатливо време за апликацијата

пред индексирање

```
EXPLAIN ANALYZE
SELECT *
FROM active_tasks_view
WHERE worker_id=95
```

AZ QUERY PLAN	
-> Parallel Seq Scan on task t	(cost=0.00..60807.36 rows=216401 width=16) (actual time=11.806..1723)
Filter: ((status)::text = 'ACTIVE'::text)	
Rows Removed by Filter: 778861	
-> Parallel Hash	(cost=571429.59..571429.59 rows=150 width=16) (actual time=988.617..988.618 row)
Buckets: 1024 Batches: 1 Memory Usage: 168kB	
-> Parallel Seq Scan on offer o	(cost=0.00..571429.59 rows=150 width=16) (actual time=482.357..5)
Execution Time: 18971.880 ms	

после индексирање:

-> Parallel Seq Scan on task t	(cost=0.00..62528.13 rows=244002 width=16) (actual time=52.667..133)
Filter: ((status)::text = 'ACTIVE'::text)	
Rows Removed by Filter: 778861	
-> Hash	(cost=2351.68..2351.68 rows=599 width=16) (actual time=1.600..1.601 rows=491 loops=5)
Buckets: 1024 Batches: 1 Memory Usage: 32kB	
-> Bitmap Heap Scan on offer o	(cost=9.08..2351.68 rows=599 width=16) (actual time=0.340..1.46)
Recheck Cond: (worker_id = 95)	
Heap Blocks: exact=484	
-> Bitmap Index Scan on idx_offer_worker	(cost=0.00..8.93 rows=599 width=0) (actual time=0.000..0.001)
Execution Time: 193.597 ms	

View 10: Task Chat View

1. Примарен филтер за погледот ќе биде според `task_id`
2. Погледот ќе се користи за прикажување на chat пораки за одреден task, заедно со информации за испраќач, улога на испраќач, task статус, worker и client.
3. Иницијалното време за извршување на query-то изнесува 226642.196ms (~3.7 минути), што е неприфатливо време за апликацијата
4. Најголем проблем во планот на извршување претставува Parallel Seq Scan on message бидејќи се обработуваат огромен број записи за да ги пронајде пораките за конкретен `task_id`
5. Query-то дополнително користи повеќе Nested Loop операции и повеќе Join операции, што ја зголемува сложеноста.
6. За оптимизација е потребен индекс:
CREATE INDEX idx_message_task ON Message(task_id);
7. Со индексирањето започна да се користи индексот и имаме Index Scan, наместо Parallel Seq Scan и времето на извршување на прашалникот изнесува 0.248ms што е прифатливо и брзо време за апликацијата

пред индексирање:

```
EXPLAIN ANALYZE
SELECT *
FROM task_chat_view
WHERE task_id = 235
```

The screenshot shows the execution plan for the query before indexing. The plan is as follows:

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Loops
1	Nested Loop	0.42..350211.21	7	89	181210.186..226517.189	0	5
2	Parallel Seq Scan on message m	0.00..350152.13	7	70	181210.048..226517.047	0	5
	Filter: (task_id = 235)						
	Rows Removed by Filter: 3758152						
3	Index Scan using useraccount_pkey on useraccount ua	0.42..8.44	1	19	0.334..0.334	1	2
	Index Cond: (id = m.sender_id)						

Planning Time: 1.732 ms
JIT:
Functions: 66
Options: Inlining false, Optimization false, Expressions true, Deforming true
Timing: Generation 5.056 ms (Deform 2.603 ms), Inlining 0.000 ms, Optimization 2.773 ms, Emission 51.842 ms, Total 59.671 ms
Execution Time: 226642.196 ms

после индексирање:

```
EXPLAIN ANALYZE
SELECT *
FROM task_chat_view
WHERE task_id = 235
```

The screenshot shows the execution plan for the query after indexing. The plan is as follows:

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Loops
	Index Cond: (id = tr.client_id)						
1	Index Scan using idx_message_task on message m	0.44..119.28	29	70	0.052..0.054	2	1
	Index Cond: (task_id = 235)						
2	Index Scan using useraccount_pkey on useraccount ua	0.42..8.44	1	19	0.006..0.006	1	2
	Index Cond: (id = m.sender_id)						

Planning Time: 1.816 ms
Execution Time: 0.248 ms

View 11: Completed Tasks View

1. Примарен филтер за погледот ќе биде според worker_id, а дополнително се користи и филтер според статус на task (COMPLETED).
2. Погледот ќе се користи за прикажување на сите завршени tasks за одреден worker, заедно со информации за client, category, location и payment summary информации.
3. Времето на извршување на query-то изнесува 29369.537ms (~29 секунди), што претставува многу бавно време за апликацијата. Најголем проблем во планот на извршување претставува Parallel Seq Scan on task, при што се скенира огромен број на записи од Task за да се пронајдат сите tasks со статус COMPLETED.
4. Дополнително query-то користи Hash Join и повеќе Nested Loop операции врз големи табели, што дополнително ја зголемува сложеноста и времето на извршување.
5. Додаден е composite индекс за оптимизација:

```
CREATE INDEX idx_task_offer_status  
ON Task(offer_id, status);
```
6. Со индексирањето, започна да се користи индексот и времето на извршување значително се намали на 128.183ms, што претставува прифатливо време за апликацијата

```
EXPLAIN ANALYZE  
SELECT *  
FROM completed_tasks_view  
where worker_id=95
```

AZ QUERY PLAN	
Hash Cond: (t.offer_id = o.id)	
-> Parallel Seq Scan on task t	(cost=0.00..66173.94 rows=785855 width=24) (actual time=1277.138..29239.949 rows=778861 loops=5)
Filter: ((status)::text = 'COMPLETED'::text)	
Rows Removed by Filter: 194715	
-> Hash	(cost=2351.68..2351.68 rows=599 width=16) (actual time=2.262..2.263 rows=491 loops=5)
Buckets: 1024 Batches: 1 Memory Usage: 32kB	
-> Bitmap Heap Scan on offer o	(cost=9.08..2351.68 rows=599 width=16) (actual time=0.280..2.093 rows=491 loops=5)
Recheck Cond: (worker_id = 95)	
Heap Blocks: exact=484	
-> Bitmap Index Scan on idx_offer_worker	(cost=0.00..8.93 rows=599 width=0) (actual time=0.156..0.156 rows=491 loops=5)
Index Cond: (worker_id = 95)	
-> Index Scan using taskrequest_pkey on taskrequest tr	(cost=0.43..8.40 rows=1 width=16) (actual time=0.013..0.013 rows=1 loops=170)
Index Cond: (id = o.task_request_id)	
-> Index Scan using client_pkey on client c	(cost=0.29..2.26 rows=1 width=8) (actual time=0.007..0.007 rows=1 loops=170)
Index Cond: (id = tr.client_id)	
-> Index Scan using useraccount_pkey on useraccount cu	(cost=0.42..2.50 rows=1 width=19) (actual time=0.008..0.008 rows=1 loops=170)
Index Cond: (id = c.user_id)	
-> Index Scan using category_pkey on category cat	(cost=0.14..0.16 rows=1 width=26) (actual time=0.003..0.003 rows=1 loops=170)
Index Cond: (id = tr.category_id)	
-> Index Scan using location_pkey on location l	(cost=0.29..2.18 rows=1 width=12) (actual time=0.003..0.003 rows=1 loops=170)
Index Cond: (id = tr.location_id)	
Planning Time: 2.632 ms	
Execution Time: 29360.537 ms	

после индексирање:

-> Index Scan using idx_task_offer_status on task t	(cost=0.43..8.37 rows=1 width=24) (actual time=0.013..0.013 rows=0 loops=491)
Index Cond: ((offer_id = o.id) AND ((status)::text = 'COMPLETED'::text))	
-> Index Scan using taskrequest_pkey on taskrequest tr	(cost=0.43..8.40 rows=1 width=16) (actual time=0.010..0.010 rows=1 loops=170)
Index Cond: (id = o.task_request_id)	
-> Index Scan using client_pkey on client c	(cost=0.29..2.26 rows=1 width=8) (actual time=0.006..0.006 rows=1 loops=170)
Index Cond: (id = tr.client_id)	
-> Index Scan using useraccount_pkey on useraccount cu	(cost=0.42..2.50 rows=1 width=19) (actual time=0.008..0.008 rows=1 loops=170)
Index Cond: (id = c.user_id)	
-> Index Scan using category_pkey on category cat	(cost=0.14..0.16 rows=1 width=26) (actual time=0.002..0.002 rows=1 loops=170)
Index Cond: (id = tr.category_id)	
Planning Time: 6.441 ms	
Execution Time: 123.183 ms	

View 12: Task Payment Summary View

1. Примарен филтер за погледот ќе биде според worker_id
2. Погледот ќе се користи за прикажување информации за плаќањата поврзани со tasks, заедно со податоците за client, worker, payment status, category и payment method.
3. Времето на извршување на query-то изнесува 155.960ms, што претставува релативно брзо извршување.
4. Во планот на извршување имаме Parallel Seq Scan on payment. Дополнително query-то користи повеќе Nested Loop операции и повеќе Index Scan операции, но сепак query-то е доволно брзо и не е потребно индексирање

```
explain ANALYZE
SELECT *
FROM task_payment_summary_view
WHERE worker_id = 95;
```

A-Z QUERY PLAN
-> Nested Loop (cost=0.43..61729.28 rows=47 width=42) (actual time=2.632..140.837 rows=39 loops=5)
-> Parallel Seq Scan on payment p (cost=0.00..61332.13 rows=47 width=38) (actual time=2.556..139.867 rows=39 loops=5)
Filter: (worker_id = 95)
Rows Removed by Filter: 973537
-> Index Scan using task_pkey on task t (cost=0.43..8.45 rows=1 width=8) (actual time=0.022..0.022 rows=1 loops=194)
Index Cond: (id = p.task_id)
-> Index Scan using client_pkey on client c (cost=0.29..6.62 rows=1 width=8) (actual time=0.012..0.012 rows=1 loops=194)
Index Cond: (id = p.client_id)
-> Index Scan using useraccount_pkey on useraccount cu (cost=0.42..2.86 rows=1 width=19) (actual time=0.011..0.011 rows=1 loops=194)
Index Cond: (id = c.user_id)
-> Index Scan using offer_pkey on offer o (cost=0.56..4.59 rows=1 width=8) (actual time=0.014..0.014 rows=1 loops=194)
Index Cond: (id = t.offer_id)
-> Index Scan using taskrequest_pkey on taskrequest tr (cost=0.43..3.00 rows=1 width=8) (actual time=0.011..0.011 rows=1 loops=194)
Index Cond: (id = o.task_request_id)
-> Hash (cost=6.13..6.13 rows=213 width=26) (actual time=0.098..0.099 rows=213 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 21kB
-> Seq Scan on category cat (cost=0.00..6.13 rows=213 width=26) (actual time=0.019..0.057 rows=213 loops=1)
Planning Time: 1.893 ms
Execution Time: 155.960 ms

View 13: Open Complaints View

1. Примарен филтер за погледот ќе биде според status, односно се прикажуваат само complaints со статус OPEN. Дополнително се користат и повеќе JOIN операции за поврзување со client, worker, task, offer, task request и category.
2. Погледот ќе се користи за прикажување на сите активни complaints во системот, заедно со информации за client, worker, task и category.
3. Иницијално време на извршување на query-то изнесување 105202ms (~105 секунди), што е неприфатливо време за апликацијата.
4. Најголем проблем со планот за извршување претставува Parallel Seq Scan on complaint, при што обработува огромен број записи за да се пронајдат само complaints со статус OPEN.
5. Дополнителни query-то користи повеќе Hash Join и Nested Loop операции брз големи табели, што дополнително ја зголемува сложеноста.
6. За оптимизација е додаден partial composite индекс:

```
CREATE INDEX idx_complaint_open
ON Complaint(task_id, client_id, worker_id)
WHERE status = 'OPEN';
```
8. Со индексирањето времето на извршување на прашалникот изнесува 4239.004ms што е прифатливо и брзо време за апликацијата
9. Беа тестирани и дополнителни индекси, но тие не донесоа значително подобрување на перформансите. Дополнително query-то користи повеќе Hash Join и Nested Loop операции врз големи табели, што дополнително ја зголемува сложеноста на планот на извршување (execution time на следните слики)
10. Вториот и третиот индекс не се користат, односно беа отстранети бидејќи не придонесуваа за подобрување на перформансите и брзината на извршување на query-то

пред индексирање:

```
EXPLAIN ANALYZE  
SELECT *  
FROM open_complaints_view
```

The screenshot shows a SQL query execution interface. At the top, there is a search bar with the text "IN ANALYZE SELECT * FROM c". Below this, there are two sections, each titled "AZ QUERY PLAN".

The first section shows a query plan with the following details:

- Hash Cond: (tr.id = o.task_request_id)
- > Parallel Seq Scan on taskrequest tr (cost=0.00..119569.75 rows=1254375 width=8) (actual time=0.467..98736.063 rows=10)
- > Parallel Hash (cost=471926.76..471926.76 rows=82983 width=126) (actual time=2378.899..2378.906 rows=66766 loops=5)
Buckets: 65536 Batches: 8 Memory Usage: 6624kB
- > Nested Loop (cost=26341.26..471926.76 rows=82983 width=126) (actual time=457.210..2288.763 rows=66766 loops=5)
-> Parallel Hash Join (cost=26340.70..102172.12 rows=82983 width=126) (actual time=457.084..827.092 rows=66766 loops=5)
Hash Cond: (t.id = comp.task_id)
- > Parallel Seq Scan on task t (cost=0.00..59493.99 rows=1217799 width=8) (actual time=0.089..129.355 rows=973)
- > Parallel Hash (cost=23015.26..23015.26 rows=107075 width=122) (actual time=98.225..98.227 rows=66766 loops=5)
Buckets: 65536 Batches: 8 Memory Usage: 6528kB
- > Parallel Seq Scan on complaint comp (cost=0.00..23015.26 rows=107075 width=122) (actual time=0.085..51.8)

The second section shows a query plan with the following details:

- Hash Cond: (cu.id = c.user_id)
- > Parallel Seq Scan on useraccount cu (cost=0.00..6021.33 rows=145833 width=19) (actual time=0.015..8.586 rows=7000)
- > Hash (cost=1443.00..1443.00 rows=100000 width=8) (actual time=26.910..26.911 rows=100000 loops=5)
Buckets: 131072 Batches: 1 Memory Usage: 4931kB
- > Seq Scan on client c (cost=0.00..1443.00 rows=100000 width=8) (actual time=0.036..9.199 rows=100000 loops=5)
- > Parallel Hash (cost=3309.59..3309.59 rows=147059 width=8) (actual time=29.328..29.329 rows=50000 loops=5)
Buckets: 262144 Batches: 1 Memory Usage: 11904kB
- > Parallel Seq Scan on worker w (cost=0.00..3309.59 rows=147059 width=8) (actual time=0.060..8.960 rows=50000 loops=5)
- > Parallel Hash (cost=6021.33..6021.33 rows=145833 width=19) (actual time=42.966..42.967 rows=70000 loops=5)
Buckets: 524288 Batches: 1 Memory Usage: 23360kB
- > Parallel Seq Scan on useraccount wu (cost=0.00..6021.33 rows=145833 width=19) (actual time=0.050..14.199 rows=70000 loops=5)

At the bottom of the first section, the execution time is shown as "Execution Time: 105202.789 ms".

после прво индексирање:

Execution Time: 4239.004 ms

после второ индексирање:

Execution Time: 4246.541 ms

после трето индексирање:

Execution Time: 4126.210 ms