Напредни бази на податоци
Фаза 3 - Индекси и оптимизација на прашалници
Проект: **MajStore**

Радица Стојкоска 231049
Теодор Трајчев 231137
Лео Тренчев 231173
* Сите мерења се однесуваат на извршувања на серверот

View1: Worker overview for his payments

1. Примарен филтер за погледот vw_worker_premium_analytics ќе биде според неговото id (worker_id), а уште ќе се користи и според месец.

2. Примарен случај на употреба ќе биде кога мајстор ќе сака да прегледа колку плаќања направил во текот на месецот, за секој план посебно . За овој поглед се важни перформансите, бидејќи бидејќи без него се губи време при извршување.

3. Иницијалното време за извршување на погледот е: **0.251s.**

```
select * from vw_worker_premium_analytics
 where worker_id = 6630371;
```

orker_premium_analytics 1 | Execution Log

ery part to search in query history

	Type	Text	Duration (ms)	Rows	Result
...	SQL / User	select * from vw_worker_premium_analytics¶where worke...	251	16	Success

Ова е прифатливо време за апликацијата па затоа нема потреба од индексирање и правење план на извршување.

4. Време на извршување при insert е: **0.265s**, на update е: **0.163s**

```
insert into Premium_Worker(start_date, end_date, plan_id, worker_id, payment_id)
values ('2024-04-05 06:58:32.962', '2024-05-05 06:58:32.962', 1, 6630496, 615);
```

tistics 1 | Execution Log

query part to search in query history

	Type	Text	Duration (ms)	Rows	Re
30 ...	SQL / User	insert into Premium_Worker(start_date, end_date, plan_id...	265	1	Su

```
update Premium_Worker set end_date='2025-05-30 15:13:01.944' where
 worker_id=6638610 and start_date= '2025-05-22 15:13:01.944';
```

tics 1 | Execution Log | Execution plan - 1 | Execution plan - 2

ery part to search in query history

	Type	Text	Duration (ms)	Rows
...	SQL / User	update Premium_Worker set end_date='2025-05-30 15:1...	163	0

View2: Owner overview for the applications revenue

1. Примарен филтер за погледот vw_owner_payment_analytics_dashboard ќе биде според датумот т.е денот а уште ќе се користи и според најкористениот метод т.е. is_most_used_method.

2. Примарен случај на употреба ќе биде кога сопственикот на апликацијата ќе сака да прегледа колку апликацијата заработила на дневно ниво, како и кој метод на плаќање највеќе го користеле корисниците и според тоа колкав процент е tax. Заработката доаѓа од плаќањата околу премиум планови.

3. Иницијалното време за извршување на погледот е: **0.537s.**

```
select * from vw_owner_payment_analytics_dashboard
where day = '2024-11-17 00:00:00.000';
```

owner_payment_analytics_dashboard 1 | Execution Log

uery part to search in query history

	Type	Text	Duration (ms)	Rows
0 ...	SQL / User	select * from vw_owner_payment_analytics_dashboard¶w...	537	1

Ова делумно е прифатливо време за апликацијата па затоа не пристапуваме кон обиди за индексирање.

5. Време на извршување при insert е: **0.105s**, на update е: **0.334s**

```
insert into Payment(amount, payment_date, worker_method_id, payment_status, plan_id)
values (120, '2026-04-30 06:58:32.962', 40056, 2, 2);
```

tatistics 1 | Execution Log

e query part to search in query history

e	Type	Text	Duration (ms)	Rows	Result
y-30 ...	SQL / User	insert into Payment(amount, payment_date, worker_meth...	105	1	Success

```
update Payment set amount=101 where
payment_date='2026-04-30 06:58:32.962' and worker_method_id=40056;
```

atistics 1 | Execution Log

query part to search in query history

	Type	Text	Duration (ms)	Rows
·30 ...	SQL / User	update Payment set amount=101 where ¶payment_date...	334	1

View3: Worker reviews per specialty

1. Примарен филтер за погледот vw_review_analytics ќе биде според worker_id и специјалност.

1. Примарен случај на употреба ќе биде кога работник на апликацијата ќе сака да прегледа колку коментари и рецензии има за неговите специјалности, како и по колку рецензии има со дадена оцена. Исто така како се користи и кога корисниците сакат да ги прегледаат рецензиите на мајсторот. За овој поглед се важни перформансите, бидејќи бидејќи без него се губи време при извршување. Оваа фунцкионалност би ја имале сите конкурентни системи па одлучивме дека е неопходна.

2. Иницијалното време за извршување на погледот е: **0.444s.**

```
select * from vw_review_analytics
where worker_id = 6630378;
```

eview_analytics 1 | Execution Log

uery part to search in query history

	Type	Text	Duration (ms)	Rows	R
...	SQL / User	select * from vw_review_analytics¶where worker_id = 663...	444	3	S

Ова е релативно прифатливо време за апликацијата па затоа не пристапуваме кон индексирање.

3. Време на извршување при insert е во табелата Review_Comment е: **0.379s** , update е: **0.226s**

```
insert into review_comment (review_id, reply)
values (3, 'The work done was not perfect.');
```

tics 1 | Execution Log | Execution plan - 1

ery part to search in query history

	Type	Text	Duration (ms)	Rows
...	SQL / User	insert into review_comment (review_id, reply)¶values (3, '...	379	1

```
update review_comment set reply='The work done was not good.'
where review_id = 3;
```

ics 1 | Execution Log | Execution plan - 1

ery part to search in query history

	Type	Text	Duration (ms)	
..	SQL / User	update review_comment set reply='The work done was n...	226	

4. Време на извршување при insert е во табелата Worker_Specialty е: **0.384s** , update е: **0.84s**

```
insert into Worker_Specialty (worker_id, specialty_id)
values (6630363, 5);
```

cs 1 | Execution Log | Execution plan - 1

ry part to search in query history

Type	Text	Duration (ms)	Rows
SQL / User	insert into Worker_Specialty (worker_id, specialty_id)¶val...	384	1

```
update Worker_Specialty set specialty_id=5
where worker_id=6630363;
```

tics 1 | Execution Log | Execution plan - 1

ery part to search in query history

Type	Text	Duration (ms)	Rows
SQL / User	update Worker_Specialty set specialty_id=5¶where worke...	84	6

View4: Worker performance overview

1. Примарен филтер на vw_worker_aggregate ќе е според worker_id

2. Примарен случај на употреба би било кога некој (клиент, најчесто работник) ќе побара на колкав број на постови аплицирал работникот, како и колкав дел од нив се прифатени / одбиени. За овој поглед се важни перформансите, бидејќи бидејќи без него се губи време при извршување.

3. Иницијалното време за извршување на погледот е: **17s**

```
select *
from vw_worker_aggregate
where worker_id = 6630359;
```

orker_aggregate 1 | Execution Log | Output

ery part to search in query history

Type	Text	Duration (ms)	Rows
SQL / User	select * ¶from vw_worker_aggregate¶where worker_id = ...	17,714	1

Ова не е прифатливо време за апликацијата па затоа пристапуваме кон индексирање.

4. Време на извршување при insert е во табелата Application е: **0.788s** , update е: **0.142s**

```
insert into application (message, needed_time, expected_price, created_at, worker_id, post_id, statu
    values ('Adding a new application', '01:00:00', 5000, '2023-06-22 14:00:03.945', 6745866, 413785
```

tistics 1 | Execution Log

query part to search in query history

	Type	Text	Duration (ms)	Rows	Result
31 ...	SQL / User	insert into application (message, needed_time, expecte...	788	1	Success

```
update Application set expected_price=5500 where application_id = 8308242;
```

tistics 1 | Execution Log

query part to search in query history

	Type	Text	Duration (ms)	Rows	Result
1 ...	SQL / User	update Application set expected_price=5500 where appl...	142	1	Success

5. Најбавните операции се: Parallel Seq Scan на табелата Application

Node Type	Entity	Cost	Rows	Time	Condition
Aggregate		445140.56 - 445145.24	1	661.404	
Sort		445140.56 - 445140.74	71	661.310	
Nested Loop		1002.51 - 445138.34	71	661.102	
Nested Loop		0.85 - 16.89	1	29.334	
Index Scan	worker	0.42 - 8.44	1	0.021	(w.worker_id = 6630359)
Index Scan	User	0.43 - 8.45	1	0.024	(u.user_id = w.user_id)
Gather		1001.66 - 445120.74	71	631.712	
Nested Loop		1.67 - 444113.54	14	586.624	
Nested Loop		1.53 - 444112.58	14	586.449	
Hash Join		1.09 - 443960.43	14	586.069	(a.status_id = apps.status_id)
Parallel Seq Scan	application	0.00 - 443959.25	14	585.858	(a.worker_id = 6630359)
Hash		1.04 - 1.04	4	0.101	
Seq Scan	application_status	0.00 - 1.04	4	0.088	
Index Scan	post	0.43 - 8.45	1	0.020	(po.post_id = a.post_id)
Memoize		0.14 - 0.16	1	0.009	
Index Scan	post_status	0.13 - 0.15	1	0.020	(ps.status_id = po.status_id)

- Истото може да се подобри со додавање индекс на worker_id во табелата Application. Прв индекс кој ќе се обидам да го ставам е B-Tree Index бидејќи истиот е ефикасен на податоци кои се пребаруваат по exact match како worker_id во случајов.

6. Времето изминато во извршување на query-то со индекс изнесува **0.082s**, што е прифатливо време на извршување.

```
create index idx_application_worker_id on application(worker_id);
```

```
select *
from vw_worker_aggregate
where worker_id = 6630378;
```

worker_aggregate 1 | Execution plan - 1 | Execution Log

uery part to search in query history

	Type	Text	Duration (ms)
2...	SQL / User	select *¶from vw_worker_aggregate¶where worker_id = 6...	82

vw_worker_aggregate 1 | Execution plan - 1 | Execution Log

Node Type	Entity	Cost	Rows	Time	Condition
Aggregate		930.13 - 934.81	1	1.093	
Sort		930.13 - 930.31	65	1.046	
Nested Loop		1.28 - 927.90	65	0.873	
Nested Loop		1.28 - 924.17	65	0.823	
Nested Loop		1.28 - 919.45	65	0.757	
Nested Loop		0.85 - 310.87	65	0.208	
Nested Loop		0.85 - 16.89	1	0.046	
Index Scan	worker	0.42 - 8.44	1	0.025	(w.worker_id = 6630378)
Index Scan	User	0.43 - 8.45	1	0.013	(u.user_id = w.user_id)
Index Scan	application	0.00 - 293.26	65	0.149	(a.worker_id = 6630378)
Index Scan	post	0.43 - 8.45	1	0.008	(po.post_id = a.post_id)
Materialize		0.00 - 1.06	2	0.001	
Seq Scan	application_status	0.00 - 1.04	4	0.017	
Materialize		0.00 - 1.04	2	0.000	
Seq Scan	post_status	0.00 - 1.03	3	0.007	

7. Време на извршување при insert е во табелата Application после индексирање е: **0.145s** , update е: **0.116s**

```
insert into application (message, needed_time, expected_price, created_at, worker_id, post_id, status_id)
    values ('Adding a new application', '01:00:00', 5000, '2023-06-22 14:00:03.945', 6722966, 4137859, 3)
```

stics 1 | Execution plan - 1 | Execution Log

uery part to search in query history

	Type	Text	Duration (ms)	Rows	Result
2...	SQL / User	insert into application (message, needed_time, expected...	145	1	Success

```
update Application set expected_price=5509 where application_id = 8308242;
```

tistics 1 | Execution plan - 1 | Execution Log

query part to search in query history

	Type	Text	Duration (ms)	Rows
7 2...	SQL / User	update Application set expected_price=5509 where appl...	116	1

View5: App performance overview

1. vw_app_aggregate содржи само една редица која што има глобални статистики за целата апликација

2. Примарен случај на употреба би било кога сопственикот на апликацијата би сакал да ги види нејзините статистики во моментот кога ја повикува функцијата, односно колку всушност постови има, колку од нив се активни, колку апликации има, колку корисници, односно дали системот е успешен.

3. Иницијалното време за извршување на погледот е: **58s**

```
select *
from vw_app_aggregate
```

app_aggregate 1 | Execution plan - 1 | Execution Log

uery part to search in query history

	Type	Text	Duration (ms)	Rows
2...	SQL / User	select *¶from vw_app_aggregate	58,223	1

Ова не е прифатливо време за апликацијата па затоа пристапуваме кон обид за индексирање.

4. Време на извршување при insert е во табелата Post е: **0.409s** , update е: **0.688s**

```
insert into post (title, description, created_at, user_id, location_id, status_id)
    values ('Needing a plumber', 'The bath is leaking', '2024-03-07 18:52:55.693', 503189, 9199, 2)
```

atistics 1 | Execution plan - 1 | Execution Log | Execution plan - 2

query part to search in query history

	Type	Text	Duration (ms)	Rows	Result
8 0...	SQL / User	¶insert into post (title, description, created_at, user_id, lo...	409	1	Success

```
update post
set description = 'The bath is leaking from the bottom'
where user_id = 503189 and created_at = '2024-03-07 18:52:55.693'
```

tics 1 | Execution plan - 1 | Execution Log | Execution plan - 2

ery part to search in query history

	Type	Text	Duration (ms)	Rows
)...	SQL / User	¶insert into post (title, description, created_at, user_id, lo...	688	1

5. Најбавните операции се: Parallel Seq Scan на Post, Application и User

tics 1 | Execution plan - 4

e Type	Entity	Cost	Rows	Time	Condition
Parallel Seq Scan	post	0.00 - 290523.26	2097894	868.220	
Gather Merge		1347449.38 - 260362...	15755763	14614.074	
Sort		1346449.32 - 135300...	3151153	10540.303	
Hash Join		395299.31 - 905449.78	3151153	7835.237	(p.status_id = ps...
Parallel Hash Join		395260.74 - 898503.31	3151153	7201.412	(p.user_id = u.us...
Parallel Hash Join		336115.59 - 804513.24	3145122	4053.899	(a.post_id = p.p...
Parallel Seq Scan	application	0.00 - 430375.14	1912002	658.900	
Parallel Hash		290523.26 - 290523.26	2097894	1314.522	
Parallel Seq Scan	post	0.00 - 290523.26	2097894	598.771	
Parallel Hash		49294.62 - 49294.62	480000	884.100	
Parallel Seq Scan	User	0.00 - 49294.62	480000	401.431	
Hash		22.70 - 22.70	3	0.067	

ct * from vw_app_aggregate

- Истото може да се подобри најпрво со додавање индекс на user_id и status_id во табелата Post. Потоа додадов индекс на created_at со desc што значи дека индексот е сортиран по опаѓачки редослед и треба да се прочита само првата редица, така што нема потреба од скенирање на целата табела. Исто така додадов индекс на post_id во Application.

6. Времето изминато во извршување на query-то со индекс изнесува **3.5s**, што не е прифатливо време на извршување

7. Следно се обидуваме да го направиме како materialized view. Од тука перформансите се подобруваат значително, односно сега извршувањето е **0.105s**, што е прифатливо време на извршување.

```
select *
from vw_app_aggregate;
```

tics 1 | Execution plan - 1 | Execution Log | Execution plan - 2

ery part to search in query history

	Type	Text	Duration (ms)	Rows
...	SQL / User	select *¶from vw_app_aggregate	105	1

View6: Статистики според локација (регион, град)

1. Примарен филтер на vw_location_aggregate ќе е според location_id, city, region.

2. Примарен случај на употреба би било кога сопственикот би сакал да ги види статистиките за работници и постови според градот или регионот. Односно да види во кои градови има најголема / најмала користеност за да прави промоции и маркетинг на апликацијата во тој регион.

3. Иницијалното време за извршување на погледот е: **2.57s**

```
select *
from vw_location_aggregate
where location_id = 1;
```

location_aggregate 1 | Execution Log | Execution plan - 2

query part to search in query history

	Type	Text	Duration (ms)	Rows	Result
31 ...	SQL / User	select * ¶from vw_location_aggregate¶where location_id ...	2,578	1	Success

Ова не е прифатливо време за апликацијата па затоа пристапуваме кон индексирање.

4. Најбавните операции се: Seq scan на Post и Worker.

Node Type	Entity	Cost	Rows	Time	Condition
Aggregate		297830.83 - 305086.87	1	568.364	
Nested Loop		297830.83 - 305083.18	658	568.126	
Gather Merge		297830.54 - 297843.11	94	537.024	
Sort		296830.48 - 296830.54	19	501.729	
Hash Join		1.07 - 296829.87	19	501.516	(p.status_id = ps.status_i
Parallel Seq Scan	post	0.00 - 296828.63	19	501.204	(p.location_id = 1)
Hash		1.03 - 1.03	3	0.079	
Seq Scan	post_status	0.00 - 1.03	3	0.065	
Materialize		0.29 - 7237.45	7	0.329	
Nested Loop		0.29 - 7237.44	7	30.914	
Index Scan	location	0.29 - 8.31	1	0.032	(l.location_id = 1)
Seq Scan	worker	0.00 - 7229.11	7	30.861	(w.location_id = 1)

- Истото може да се подобри со додавање индекс на location_id во табелата Worker и Post . Прв индекс кој ќе се обидам да го ставам само на табелата Post е B-Tree Index бидејќи истиот е ефикасен на податоци кои се пребаруваат по exact match како location_id во случајов.

5. Време на извршување при insert е во табелата Post е: **0.304s** , update е: **0.137s**

```
insert into post(title, description, created_at, user_id, location_id, status_id)
    values ('Roof leaking', 'Roof leaking', '2025-04-23 10:12:08.179', 102498, 44603, 1)
```

atistics 1 | Execution Log | Execution plan - 2 | Execution plan - 3

query part to search in query history

	Type	Text	Duration (ms)	Rows	Result
31 ...	SQL / User	insert into post(title, description, created_at, user_id, loca...	304	1	Success

```
update post set title='Roof leak' where post_id = 6219906
```

tics 1 | Execution Log | Execution plan - 2 | Execution plan - 3

ery part to search in query history

	Type	Text	Duration (ms)	Rows
..	SQL / User	update post set title='Roof leak' where post_id = 621990...	137	1

6. Времето изминато во извршување на query-то со индекс изнесува **0.110s**, што е прифатливо време на извршување.

```
create index idx_post_location on Post(location_id);
```

```
select *
from vw_location_aggregate
where location_id = 7;
```

ocation_aggregate 1 | Execution plan - 1 | Execution Log | Execution plan - 2 | Execution plan - 3

uery part to search in query history

	Type	Text	Duration (ms)	Rows
0...	SQL / User	select *¶from vw_location_aggregate¶where location_id ...	110	1

vw_location_aggregate 1 | Execution plan - 1 | Execution Log | Execution plan - 2 | Execution plan - 3

Node Type	Entity	Cost	Rows	Time	Condition
Aggregate		7673.68 - 7677.89	1	30.471	
Sort		7673.68 - 7674.21	101	30.418	
Nested Loop		5.54 - 7665.58	101	30.364	
Nested Loop		0.29 - 7237.44	1	30.016	
Index Scan	location	0.29 - 8.31	1	0.027	(l.location_id = 7)
Seq Scan	worker	0.00 - 7229.11	1	29.986	(w.location_id = 7)
Materialize		5.25 - 425.78	101	0.326	
Nested Loop		5.25 - 425.25	101	0.281	
Bitmap Heap Scan	post	5.25 - 420.28	101	0.200	
Bitmap Index Scan	idx_post_location	0.00 - 5.22	101	0.018	(p.location_id = 7)
Materialize		0.00 - 1.04	2	0.000	
Seq Scan	post_status	0.00 - 1.03	3	0.008	

View7: Post full details

1. Примарен филтер за погледот vw_post_full_details ќе биде според неговото id (post_id), а уште ќе се користи и според статусот на огласот и корисничкото име на креаторот.

2. Примарен случај на употреба ќе биде кога корисникот ќе сака да прегледа детали за одреден оглас т.е кој го креирал, на која локација е и статистики од апликациите (вкупно, просечна/минимална/максимална очекувана цена), сликите кои ги поставил, кои специјалности се потребни. За овој поглед се важни перформансите, бидејќи без него се губи време при извршување на комплексен query со повеќе joins.
3. Иницијалното време за извршување на погледот е: **5,649s**.

```
select *
from vw_post_full_details
where post_id = 1539670;
```

post_full_details 1 | Execution plan - 1 | Execution Log

query part to search in query history

	Type	Text	Duration (ms)	Rows
1...	SQL / User	select *¶from vw_post_full_details¶where post_id = 1539...	5,649	1

Ова не е прифатливо време за апликацијата па затоа пристапуваме кон индексирање.

4. Најбавните операции се: Parallel Seq Scan на табелата Application, Post_Specialty и Post_Image.

vw_post_full_details 1 | Execution plan - 1

Node Type	Entity	Cost	Rows	Time	Condition
Nested Loop		3001.29 - 787046.77	10	1742.264	
Nested Loop		3001.29 - 787040.21	10	1742.163	
Nested Loop		2001.29 - 577440.09	10	1303.748	
Nested Loop		1001.29 - 444999.81	5	1174.657	
Nested Loop		1.29 - 37.43	1	910.441	
Nested Loop		0.99 - 29.12	1	910.405	
Nested Loop		0.56 - 20.68	1	910.337	
Index Scan	post_status	0.13 - 12.18	3	0.040	
Materialize		0.43 - 8.46	1	303.419	
Index Scan	User	0.43 - 8.45	1	0.045	(u.user_id = p.user_id)
Index Scan	location	0.29 - 8.31	1	0.018	(l.location_id = p.location_id)
Gather		1000.00 - 444962.34	5	264.200	
Parallel Seq Scan	application	0.00 - 443962.04	1	674.493	(a.post_id = 1539670)
Materialize		1000.00 - 132440.14	2	25.813	
Gather		1000.00 - 132440.12	2	129.049	
Nested Loop		0.00 - 131439.73	0	446.157	
Parallel Seq Scan	post_specialty	0.00 - 131437.04	0	446.135	(psp.post_id = 1539670)
Seq Scan	specialty	0.00 - 1.75	52	0.015	
Materialize		1000.00 - 209598.80	1	43.840	
Gather		1000.00 - 209598.75	1	438.382	
Parallel Seq Scan	post_image	0.00 - 208597.85	0	406.129	(pi.post_id = 1539670)
Materialize		0.00 - 1.06	3	0.007	
Seq Scan	application_status	0.00 - 1.04	4	0.060	

- Истото може да се подобри со додавање B-tree индекс на post_id во табелите Application, Post_Specialty и Post_Image. B-tree индексот е ефикасен на податоци каде треба да се најде exact match.

5. Време на извршување при insert е во табелата Post_Specialty е: **0.155s**, update: **1.526s.**

```
insert into Post_Specialty(post_id, specialty_id)
    values (919803, 1);
```

stics 1 | Execution Log | Execution plan - 1

ery part to search in query history

	Type	Text	Duration (ms)	Rows
0...	SQL / User	insert into Post_Specialty(post_id, specialty_id)¶ values (9...	155	1

```
update Post_Specialty
set specialty_id = 2
where post_id = 1539671;
```

ost_full_details 1 | Statistics 2 | Execution Log | Execution plan - 1

ery part to search in query history

	Type	Text	Duration (ms)	Rows	Re:
0...	SQL / User	update Post_Specialty ¶set specialty_id = 2¶where post_i...	1,526	2	Su

6. Време на извршување при insert е во табелата Post_Image е: **0.3s**, update: **1.189s.**

```
insert into Post_Image(image_file, post_id)
    values ('x48562878a554caab5dd263b0e66a3ff246d272d4988137609211e640faddb168967b0a
```

stics 1 | Execution Log | Execution plan - 1

ery part to search in query history

	Type	Text	Duration (ms)	Rows
0...	SQL / User	insert into Post_Image(image_file, post_id)¶ values ('x485...	300	1

```
update Post_Image
set image_file = 'x48562878a554caab5dd263b0e66a3ff246d272d4988137609211e640faddb168967b0ae78a446bebbd26b6c6386d548e8f38474363c68595cf054f3a21
where post_id = 1863170;
```

_post_full_details 1 | Statistics 2 | Execution Log | Execution plan - 1

query part to search in query history

	Type	Text	Duration (ms)	Rows	Result
1 0...	SQL / User	update Post_Image¶set image_file = 'x48562878a554caa...	1,189	0	Success

7. Времето изминато во извршување на query-то со индекс изнесува **0.087s**, што е прифатливо време на извршување,

```
create index idx_application_post_id on Application(post_id);

create index idx_post_specialty_post_id on Post_Specialty(post_id);

create index idx_post_image_post_id on Post_Image(post_id);
```

```
select *
from vw_post_full_details
where post_id = 1539670;
```

post_full_details 1 | Execution Log | Execution plan - 1

query part to search in query history

	Type	Text	Duration (ms)	Rows
0...	SQL / User	select *¶from vw_post_full_details¶where post_id = 1539...	87	1

vw_post_full_details 1 | Execution Log | Execution plan - 1

Node Type	Entity	Cost	Rows	Time	Condition
Aggregate		69.44 - 76.19	1	0.312	
Sort		69.44 - 69.71	10	0.255	
Hash Join		6.23 - 65.79	10	0.208	(a.status_id = apps.status_id)
Nested Loop		5.14 - 64.12	10	0.185	
Nested Loop		4.71 - 54.16	10	0.164	
Hash Join		4.28 - 37.52	2	0.131	(psp.specialty_id = s.specialty_id)
Nested Loop		1.59 - 34.82	2	0.081	
Nested Loop		1.16 - 26.28	1	0.063	
Nested Loop		0.86 - 17.97	1	0.053	
Nested Loop		0.43 - 9.52	1	0.039	
Index Scan	User	0.43 - 8.45	1	0.011	(u.user_id = p.user_id)
Index Scan	location	0.29 - 8.31	1	0.008	(l.location_id = p.location_id)
Index Scan	post_specialty	0.43 - 8.51	2	0.016	(psp.post_id = 1539670)
Hash		1.75 - 1.75	75	0.039	
Seq Scan	specialty	0.00 - 1.75	75	0.020	
Materialize		0.43 - 16.50	5	0.014	
Index Scan	application	0.43 - 16.48	5	0.021	(a.post_id = 1539670)
Materialize		0.43 - 8.63	1	0.002	
Index Scan	post_image	0.43 - 8.59	1	0.014	(pi.post_id = 1539670)
Hash		1.04 - 1.04	4	0.012	
Seq Scan	application_status	0.00 - 1.04	4	0.009	

8. Време на извршување при update е во табелата Post_Specialty после индексирање е: **0.02s.**

```
update Post_Specialty
set specialty_id = 2
where post_id = 1539671;
```

tistics 1 | Execution Log

query part to search in query history

	Type	Text	Duration (ms)	Rows	Result
0...	SQL / User	update Post_Specialty ¶set specialty_id = 2¶where post_i...	20	2	Success

9. Време на извршување при update е во табелата Post_Image после индексирање е: **0.026s.**

```
update Post_Image
set image_file = 'x48562878a554caab5dd263b0e66a3ff246d272d4988137609211e640faddb168967b0ae78a446bebbd26b6c6386d548e8f38474
where post_id = 1863170;
```

atistics 1 | Execution Log

query part to search in query history

	Type	Text	Duration (ms)	Rows	Result
11 0...	SQL / User	update Post_Image¶set image_file = 'x48562878a554caa...	26	0	Success

Дополнително бидејќи овој view користи 3 индекси, пробавме да го направиме materialized, но така перформансите не беа толку добри односно 0.556s па сепак view-то останува индексирано

View8: User notification

2. Примарен филтер за погледот vw_user_notification ќе биде според неговото id (user_id).

3. Примарен случај на употреба ќе биде кога корисник на аплиакцијата (без разлика дали е обичен корисник или работник) ќе сака да ги прегледа нотификациите кои му пристигнале од апликацијата, без разлика дали се прочитани или не. Оваа фунцкионалност би ја имале сите конкурентни системи па одлучивме дека е неопходна.

4. Иницијалното време за извршување на погледот е: **0.108s**.

```
select *
from vw_user_notifications
where user_id = 210032;
```

tics 1 | Execution Log

ery part to search in query history

	Type	Text	Duration (ms)	Rows	Result
..	SQL / User	select *¶from vw_user_notifications ¶where user_id = 210...	108	3	Success

Ова е прифатливо време за апликацијата

Statistics 1 | Execution Log | Execution plan - 1

Node Type	Entity	Cost	Rows	Time	Condition
Gather Merge		6020.40 - 6020.51	3	386.254	
Sort		5020.39 - 5020.39	2	15.121	
Nested Loop		0.43 - 5020.38	2	15.071	
Parallel Seq Scan	notification	0.00 - 5011.92	2	15.051	(n.user_id = 210032)
Index Scan	User	0.43 - 8.45	1	0.011	(u.user_id = 210032)

5. Време на извршување при insert е во табелата Notification е: **0.19s**, update: **0.110s.**

```
insert into notification(message, is_read, created_at, user_id)
    values ('New worker has sent you an application', false, '2026-03-13 02:20:26.203', 210032)
```

stics 1 | Execution Log

iery part to search in query history

	Type	Text	Duration (ms)	Rows	Result
1...	SQL / User	insert into notification(message, is_read, created_at, user...	19	1	Success

```
update notification
set is_read = true
where user_id = 210032 and created_at = '2026-03-13 02:20:26.203'
```

tistics 1 | Execution Log

query part to search in query history

	Type	Text	Duration (ms)	Rows
' 1...	SQL / User	update notification¶set is_read = true ¶where user_id = 2...	110	1

View9: Worker applications overview

1. Примарен филтер за погледот vw_worker_applications ќе биде според неговото id (worker_id), а уште ќе се користи и според статусот на апликацијата.

2. Примарен случај на употреба ќе биде кога мајстор ќе сака да ги прегледа сите свои апликации - за кои огласи аплицирал, каков е статусот на секоја апликација (Pending, Accepted, Rejected), на која локација е огласот и по каква цена аплицирал. За овој поглед се важни перформансите, бидејќи без него се губи време при извршување.

3. Иницијалното време за извршување на погледот е: **0.147s**.

```
select *
from vw_worker_applications
where worker_id = 6630371
```

worker_applications 1 | Execution plan - 1 | Execution Log | Execution plan - 2 | Execution plan - 3

query part to search in query history

	Type	Text	Duration (ms)	Rows
0...	SQL / User	select * ¶from vw_worker_applications¶where worker_id ...	147	79

Ова е прифатливо време за апликацијата па затоа не пристапуваме кон индексирање.

vw_worker_applications 1 | Execution plan - 1 | Execution Log | Execution plan - 2 | Execution plan - 3 | Execution plan - 4

Node Type	Entity	Cost	Rows	Time	Condition
Nested Loop		1.58 - 1109.59	79	1453.806	
Nested Loop		1.28 - 927.90	79	1147.344	
Nested Loop		1.28 - 924.17	79	1147.259	
Nested Loop		1.28 - 919.45	79	1147.092	
Nested Loop		0.85 - 310.87	79	31.874	
Nested Loop		0.85 - 16.89	1	14.400	
Index Scan	worker	0.42 - 8.44	1	0.019	(w.worker_id = 6630371)
Index Scan	User	0.43 - 8.45	1	14.374	(u.user_id = w.user_id)
Index Scan	application	0.00 - 293.26	79	17.436	(a.worker_id = 6630371)
Index Scan	post	0.43 - 8.45	1	14.114	(p.post_id = a.post_id)
Materialize		0.00 - 1.06	3	0.001	
Seq Scan	application_status	0.00 - 1.04	4	0.012	
Materialize		0.00 - 1.04	2	0.000	
Seq Scan	post_status	0.00 - 1.03	3	0.007	
Index Scan	location	0.29 - 2.52	1	3.877	(l.location_id = p.location_id)

Овој поглед дополнително го користи претходно поставениот индекс на View4 (idx_application_worker_id) на табелата Application (worker_id)