

Напредни бази на податоци

Фаза 3: Оптимизација на прашалници и погледи

Проект: PawCare

Ѓургица Танева 231053
Благоица Јосифоска 231039

View1: Daily clinic schedule

1. Примарен филтер за погледот daily_clinic_schedule ќе биде според датум на закажан термин (date_appointment) или според id на одговорен ветеринар (employee_id).
2. Кога клиниката сака да добие преглед врз закажаните термини за даден датум и кога одреден вработен сака да добие дневен распоред за работа. За овој поглед ни се важни перформансите, бидејќи наменет е за секојдневна употреба.
3. Иницијалното време за извршување на погледот е 560ms (192ms execution, 368ms fetching) кога се пребаруваат термини за одреден датум и 537ms (183ms execution, 354ms fetching) кога вработен пребарува свои термини за даден датум.
4. Најбавните операции се full sequential scans на табелите appointment и examination.

```
[2026-09-09 11:45:51] vet_db.public> SELECT * FROM daily_clinic_schedule WHERE date_appointment = '2025-07-07'  
[2026-09-09 11:45:52] 442 rows retrieved starting from 1 in 560 ms (execution: 192 ms, fetching: 368 ms)
```

```
[2026-09-09 12:02:35] vet_db.public> SELECT * FROM daily_clinic_schedule  
WHERE employee_id = 3 AND date_appointment = '2025-07-07'  
[2026-09-09 12:02:36] 50 rows retrieved starting from 1 in 537 ms (execution: 183 ms, fetching: 354 ms)
```

QUERY PLAN	
1	Gather (cost=15045.23..30106.75 rows=464 width=1705) (actual time=83.538..158.236 rows=442 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	Buffers: shared hit=3652 read=17323
5	-> Nested Loop Left Join (cost=14045.23..29860.35 rows=193 width=1705) (actual time=39.250..107.658 rows=147 loops=3)
6	Buffers: shared hit=3652 read=17323
7	-> Hash Join (cost=14045.08..29852.25 rows=193 width=635) (actual time=39.222..107.273 rows=147 loops=3)
8	Hash Cond: (a.pet_id = p.id)
9	Buffers: shared hit=3602 read=17323
10	-> Hash Join (cost=13837.24..28843.90 rows=193 width=622) (actual time=36.460..104.369 rows=147 loops=3)
11	Hash Cond: (a.owner_id = o.id)
12	Buffers: shared hit=3350 read=17323
13	-> Hash Left Join (cost=13757.29..28763.44 rows=193 width=593) (actual time=35.237..103.021 rows=147 loops=3)
14	Hash Cond: (e.employee_id = emp.id)
15	Buffers: shared hit=3245 read=17323
16	-> Parallel Hash Right Join (cost=13746.50..28750.00 rows=193 width=64) (actual time=35.008..102.684 rows=147 loops=3)
17	Hash Cond: (e.appointment_id = a.id)
18	Buffers: shared hit=3210 read=17323
19	-> Parallel Seq Scan on examination e (cost=0.00..14216.00 rows=300000 width=26) (actual time=0.243..23.487 rows=240000 loops=3)
20	Buffers: shared hit=1598 read=9618
21	-> Parallel Hash (cost=13744.08..13744.08 rows=193 width=42) (actual time=34.392..34.392 rows=147 loops=3)
22	Buckets: 1024 Batches: 1 Memory Usage: 104kB
23	Buffers: shared hit=1612 read=7705
24	-> Parallel Seq Scan on appointment a (cost=0.00..13744.08 rows=193 width=42) (actual time=0.527..34.180 rows=147 loops=3)
25	Filter: (date_appointment = '2025-07-07'::date)
26	Rows Removed by Filter: 283186
27	Buffers: shared hit=1612 read=7705
28	-> Hash (cost=10.16..10.16 rows=50 width=533) (actual time=0.717..0.716 rows=50 loops=3)
29	Buckets: 1024 Batches: 1 Memory Usage: 12kB

QUERY PLAN	
1	Gather (cost=14747.35..29851.91 rows=42 width=1705) (actual time=70.506..113.990 rows=50.00 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	Buffers: shared hit=4049 read=16853
5	-> Nested Loop Left Join (cost=13747.35..28847.71 rows=18 width=1705) (actual time=29.031..67.697 rows=16.67 loops=3)
6	Buffers: shared hit=4049 read=16853
7	-> Nested Loop (cost=13747.20..28845.74 rows=18 width=635) (actual time=29.006..67.594 rows=16.67 loops=3)
8	Buffers: shared hit=4001 read=16853
9	-> Nested Loop (cost=13746.92..28822.49 rows=18 width=622) (actual time=28.986..67.440 rows=16.67 loops=3)
10	Buffers: shared hit=3849 read=16853
11	-> Nested Loop (cost=13746.64..28809.73 rows=18 width=593) (actual time=28.964..67.255 rows=16.67 loops=3)
12	Buffers: shared hit=3697 read=16853
13	-> Parallel Hash Join (cost=13746.50..28797.49 rows=18 width=64) (actual time=28.848..67.099 rows=16.67 loops=3)
14	Hash Cond: (e.appointment_id = a.id)
15	Buffers: shared hit=3680 read=16853
16	-> Parallel Seq Scan on examination e (cost=0.00..14966.00 rows=32380 width=26) (actual time=0.199..37.668 rows=26743.67 loops=3)
17	Filter: (employee_id = 3)
18	Rows Removed by Filter: 213256
19	Buffers: shared hit=1880 read=9336
20	-> Parallel Hash (cost=13744.08..13744.08 rows=193 width=42) (actual time=25.864..25.865 rows=147.33 loops=3)
21	Buckets: 1024 Batches: 1 Memory Usage: 72kB
22	Buffers: shared hit=1800 read=7517
23	-> Parallel Seq Scan on appointment a (cost=0.00..13744.08 rows=193 width=42) (actual time=0.221..25.731 rows=147.33 loops=3)
24	Filter: (date_appointment = '2025-07-07'::date)
25	Rows Removed by Filter: 283186
26	Buffers: shared hit=1800 read=7517
27	-> Materialize (cost=0.14..12.02 rows=1 width=533) (actual time=0.007..0.008 rows=1.00 loops=50)
28	Storage: Memory Maximum Storage: 17kB
29	Buffers: shared hit=17

Времето изминато во извршување на операциите insert и update изнесува:

```
[2026-09-09 13:15:04] vet_db.public> INSERT INTO appointment (date_appointment, reason, phone, owner_id, pet_id)
VALUES ('2025-07-07', 'Routine checkup', '+389 71 271 919', 1, 385)
[2026-09-09 13:15:04] 1 row affected in 24 ms
[2026-09-09 13:15:42] vet_db.public> UPDATE appointment
SET reason = 'Follow-up visit'
WHERE date_appointment = '2025-07-07' AND owner_id = 1
[2026-09-09 13:15:42] 2 rows affected in 90 ms

[2026-09-09 13:21:16] vet_db.public> INSERT INTO examination (date_examination, status, appointment_id, employee_id, examination_room_id)
VALUES ('2025-07-10', 'scheduled', 850801, 3, 5)
[2026-09-09 13:21:16] 1 row affected in 17 ms
[2026-09-09 13:21:46] vet_db.public> UPDATE examination
SET status = 'completed'
WHERE employee_id = 3 AND date_examination = '2025-07-10'
[2026-09-09 13:21:46] 43 rows affected in 114 ms
```

```
CREATE INDEX idx_appointment_date ON appointment (date_appointment);
CREATE INDEX idx_examination_employee ON examination (employee_id);
CREATE INDEX idx_examination_appointment ON examination (appointment_id);
```

5. Времето изминато во извршување на query-то со индекси изнесува 414ms (22ms execution, 392ms fetching) и 396ms (<1ms execution, 396ms fetching). По индексирањето, времето на извршување (execution) значително се намали, времето на преземање на податоците (fetching) остана непроменето, бидејќи тоа зависи од бројот на вратени редици (443) и нивната ширина, а не од индексирањето.

QUERY PLAN

12	-> Hash Join (cost=87.98..1607.33 rows=464 width=71) (actual time=1.169..2.186 rows=443.00 loops=1)
13	Hash Cond: (a.owner_id = o.id)
14	Buffers: shared hit=469
15	-> Bitmap Heap Scan on appointment a (cost=8.02..1526.16 rows=464 width=42) (actual time=0.129..0.739 rows=443.00 loops=1)
16	Recheck Cond: (date_appointment = '2025-07-07'::date)
17	Heap Blocks: exact=430
18	Buffers: shared hit=434
19	-> Bitmap Index Scan on idx_appointment_date (cost=0.00..7.91 rows=464 width=0) (actual time=0.062..0.062 rows=443.00 loops=1)
20	Index Cond: (date_appointment = '2025-07-07'::date)
21	Index Searches: 1
22	Buffers: shared hit=4
23	-> Hash (cost=54.98..54.98 rows=1998 width=33) (actual time=1.029..1.029 rows=1998.00 loops=1)
24	Buckets: 2048 Batches: 1 Memory Usage: 146kB
25	Buffers: shared hit=35
26	-> Seq Scan on owner o (cost=0.00..54.98 rows=1998 width=33) (actual time=0.010..0.467 rows=1998.00 loops=1)
27	Buffers: shared hit=35
28	-> Hash (cost=139.04..139.04 rows=5504 width=17) (actual time=2.429..2.429 rows=5504.00 loops=1)
29	Buckets: 8192 Batches: 1 Memory Usage: 333kB
30	Buffers: shared hit=84
31	-> Seq Scan on pet p (cost=0.00..139.04 rows=5504 width=17) (actual time=0.014..1.035 rows=5504.00 loops=1)
32	Buffers: shared hit=84
33	-> Index Scan using idx_examination_appointment on examination e (cost=0.42..7.95 rows=1 width=26) (actual time=0.004..0.004 rows=0.84 loops=443)
34	Index Cond: (appointment_id = a.id)
35	Index Searches: 443
36	Buffers: shared hit=1701
37	-> Memoize (cost=0.15..0.17 rows=1 width=21) (actual time=0.000..0.000 rows=0.84 loops=443)
38	Cache Key: e.employee_id
39	Cache Mode: logical
40	Hits: 433 Misses: 10 Evictions: 0 Overflows: 0 Memory Usage: 2kB

QUERY PLAN

16	Buffers: shared hit=2441
17	-> Nested Loop (cost=8.72..5251.05 rows=42 width=93) (actual time=0.125..2.181 rows=51.00 loops=1)
18	Buffers: shared hit=2288
19	-> Nested Loop (cost=8.45..5221.28 rows=42 width=64) (actual time=0.120..2.057 rows=51.00 loops=1)
20	Buffers: shared hit=2135
21	-> Bitmap Heap Scan on appointment a (cost=8.02..1526.16 rows=464 width=42) (actual time=0.088..0.483 rows=443.00 loops=1)
22	Recheck Cond: (date_appointment = '2025-07-07'::date)
23	Heap Blocks: exact=430
24	Buffers: shared hit=434
25	-> Bitmap Index Scan on idx_appointment_date (cost=0.00..7.91 rows=464 width=0) (actual time=0.042..0.042 rows=443.00 loops=1)
26	Index Cond: (date_appointment = '2025-07-07'::date)
27	Index Searches: 1
28	Buffers: shared hit=4
29	-> Index Scan using idx_examination_appointment on examination e (cost=0.42..7.95 rows=1 width=26) (actual time=0.003..0.003 rows=0.12 loops=443)
30	Index Cond: (appointment_id = a.id)
31	Filter: (employee_id = 3)
32	Rows Removed by Filter: 1
33	Index Searches: 443
34	Buffers: shared hit=1701
35	-> Index Scan using owner_pkey on owner o (cost=0.28..0.71 rows=1 width=33) (actual time=0.002..0.002 rows=1.00 loops=51)
36	Index Cond: (id = a.owner_id)
37	Index Searches: 51

Забелешка: Кај вториот прашалник (филтрирање по датум и вработен) `idx_examination_employee` не се користи, планерот наместо тоа го применува `employee_id` како преостанат филтер по `join`-от преку `appointment_id`, бидејќи множеството резултати филтрирано по датум е веќе доволно мало (443 редици) и не е потребно дополнително пребарување преку индекс. Овој индекс би бил корисен за прашалници кои филтрираат само по `employee_id`, без ограничување по датум, што не е идејата на овој поглед така што индексот го отстрануваме.

6. Времето изминато во извршување на операциите insert и update по индексирање изнесува:

```
[2026-09-09 14:18:14] vet_db.public> INSERT INTO appointment (date_appointment, reason, phone, owner_id, pet_id)
VALUES ('2025-07-08', 'Vaccination check', '+389 77 979 331', 2, 230)
[2026-09-09 14:18:14] 1 row affected in 9 ms
[2026-09-09 14:19:15] vet_db.public> UPDATE appointment
SET reason = 'Vaccination follow-up visit'
WHERE date_appointment = '2025-07-08' AND owner_id = 2
[2026-09-09 14:19:15] 1 row affected in 4 ms

[2026-09-09 14:21:34] vet_db.public> INSERT INTO examination (date_examination, status, appointment_id, employee_id, examination_room_id)
VALUES ('2025-07-15', 'scheduled', 850802, 3, 5)
[2026-09-09 14:21:34] 1 row affected in 3 ms
[2026-09-09 14:21:58] vet_db.public> UPDATE examination
SET status = 'completed'
WHERE employee_id = 3 AND date_examination = '2025-07-15'
[2026-09-09 14:21:58] 40 rows affected in 55 ms
```

View2: Pet medical history

1. Примарен филтер за погледот pet_medical_history ќе биде според id на милениче, а уште ќе се користи и според id на сопственик и датум на одржан термин.
2. Примарен случај на употреба ќе е пребарување на целосната здравствена историја за едно милениче при посета на клиниката: appointments, examinations и treatments, вклучувајќи ги сите type-specific атрибути како JSON за секој тип treatment (consultation/vaccination/prescription/operation) како резултат на EAV шемата што ја следи со различно множество на атрибути за секој од нив. За овој поглед ни се важни перформансите, бидејќи без него се губи време при извршување.
3. Иницијалното време на извршување на прашалникот е 48s 133ms. Ова не е прифатливо време за апликацијата па затоа пристапуваме кон индексирање.

```
[2026-09-10 13:45:11] vet_db.public> SELECT * FROM pet_medical_history WHERE pet_id = 9956
[2026-09-10 13:45:59] 154 rows retrieved starting from 1 in 48 s 133 ms (execution: 47 s 556 ms, fetching: 577 ms)
```

4. Најбавната операција е подпрашалникот кој го гради treatment_details: за секој од 154-те третмани на миленикот, табелата treatment_attribute_value целосно се скенира секвенцијално (Seq Scan), при што се отфрлаат 2.83 милиони редици што не се совпаѓаат (по секое скенирање), а тоа се извршува еднаш за секој ред од treatment.

64	QUERY PLAN
65	Buffers: shared hit=15
66	Worker 0: Hits: 48 Misses: 3 Evictions: 0 Overflows: 0 Memory Usage: 1kB
67	-> Index Scan using treatment_type_pkey on treatment_type tt (cost=0.14..0.16 rows=1 width=520) (actual time=0.012..0.012 rows=0.88 loops=8)
68	Index Cond: (id = t.treatment_type_id)
69	Index Searches: 7
70	Buffers: shared hit=15
71	SubPlan 1
72	-> Aggregate (cost=54693.54..54693.55 rows=1 width=32) (actual time=291.143..291.143 rows=1.00 loops=154)
73	Buffers: shared hit=14630 read=2952488
74	-> Nested Loop (cost=0.00..54693.53 rows=5 width=527) (actual time=232.924..291.121 rows=3.42 loops=154)
75	Join Filter: (ta.id = tav.treatment_attribute_id)
76	Rows Removed by Join Filter: 85
77	Buffers: shared hit=14630 read=2952488
78	-> Seq Scan on treatment_attribute ta (cost=0.00..10.70 rows=70 width=520) (actual time=0.002..0.004 rows=26.00 loops=154)
79	Buffers: shared hit=154
80	-> Materialize (cost=0.00..54677.59 rows=5 width=15) (actual time=0.744..11.196 rows=3.42 loops=4004)
81	Storage: Memory Maximum Storage: 17kB
82	Buffers: shared hit=14476 read=2952488
83	-> Seq Scan on treatment_attribute_value tav (cost=0.00..54677.56 rows=5 width=15) (actual time=227.322..291.080 rows=3.42 loops=154)
84	Filter: (treatment_id = t.id)
85	Rows Removed by Filter: 2832922
86	Buffers: shared hit=14476 read=2952488
87	Planning:
88	Buffers: shared hit=42
89	Planning Time: 2.109 ms
90	Execution Time: 44994.267 ms

Времето изминато во извршување на операциите insert и update изнесува:

```
[2026-09-10 14:11:46] vet_db.public> INSERT INTO treatment (examination_id, treatment_type_id, date_treatment, notes)
VALUES (1726, 2, CURRENT_DATE, 'Follow-up dosage adjustment')
[2026-09-10 14:11:46] 1 row affected in 24 ms
[2026-09-10 14:12:20] vet_db.public> INSERT INTO treatment_attribute_value (treatment_id, treatment_attribute_id, value)
VALUES (577416, 13, 'false')
[2026-09-10 14:12:20] 1 row affected in 14 ms
```

```
[2026-09-10 14:21:11] vet_db.public> UPDATE treatment
SET notes = 'Updated post-op notes'
WHERE id = 1577
[2026-09-10 14:21:11] 1 row affected in 9 ms
[2026-09-10 14:21:11] vet_db.public> UPDATE treatment_attribute_value
SET value = '122'
WHERE treatment_id = 1577 AND treatment_attribute_id = 23
[2026-09-10 14:21:11] 1 row affected in 253 ms
```

```
CREATE INDEX idx_appointment_pet ON appointment (pet_id);
CREATE INDEX idx_treatment_attribute_value_treatment ON treatment_attribute_value (treatment_id);
```

```
[2026-09-10 14:31:11] vet_db.public> SELECT * FROM pet_medical_history WHERE pet_id = 9956
[2026-09-10 14:31:11] 154 rows retrieved starting from 1 in 429 ms (execution: 76 ms, fetching: 353 ms)
```

5. Времето изминато во извршување на query-то со индекси изнесува 429ms и тоа е прифатливо време.

```

-> Hash Right Join (cost=1835.95..17029.17 rows=154 width=192) (actual time=93.532..93.765 rows=154.00 loops=1)
  Hash Cond: (t.examination_id = e.id)
  Buffers: shared hit=1542 read=6329
  -> Seq Scan on treatment t (cost=0.00..13032.27 rows=575927 width=72) (actual time=0.020..0.348 rows=575849.00 loops=1)
    Buffers: shared hit=944 read=6329
  -> Hash (cost=1834.02..1834.02 rows=154 width=124) (actual time=0.429..0.430 rows=154.00 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 31kB
    Buffers: shared hit=598
    -> Nested Loop Left Join (cost=0.85..1834.02 rows=154 width=124) (actual time=0.017..0.384 rows=154.00 loops=1)
      Buffers: shared hit=598
      -> Index Scan using idx_appointment_pet on appointment a (cost=0.42..556.34 rows=154 width=38) (actual time=0.010..0.034 rows=154.00 loops=1)
        Index Cond: (pet_id = 9956)
        Index Searches: 1
        Buffers: shared hit=6
      -> Index Scan using idx_examination_appointment on examination e (cost=0.42..0.29 rows=1 width=90) (actual time=0.002..0.002 rows=0.84 loops=154)
        Index Cond: (appointment_id = a.id)
        Index Searches: 154
        Buffers: shared hit=592

61 SubPlan 1
62 -> Aggregate (cost=32.76..32.77 rows=1 width=32) (actual time=0.005..0.005 rows=1.00 loops=154)
63   Buffers: shared hit=422
64   -> Hash Join (cost=12.00..32.75 rows=5 width=527) (actual time=0.002..0.004 rows=3.42 loops=154)
65     Hash Cond: (tav.treatment_attribute_id = ta.id)
66     Buffers: shared hit=422
67     -> Index Scan using idx_treatment_attribute_value_treatment on treatment_attribute_value tav (cost=0.43..21.16 rows=5 width=15) (actual time=0.001..0.002 rows=3.42 loops=154)
68       Index Cond: (treatment_id = t.id)
69       Index Searches: 104
70       Buffers: shared hit=421
71     -> Hash (cost=10.70..10.70 rows=70 width=520) (actual time=0.018..0.018 rows=26.00 loops=1)
72       Buckets: 1024 Batches: 1 Memory Usage: 10kB
73       Buffers: shared hit=1
74     -> Seq Scan on treatment_attribute ta (cost=0.00..10.70 rows=70 width=520) (actual time=0.009..0.012 rows=26.00 loops=1)
75       Buffers: shared hit=1
76 Planning:
77   Buffers: shared hit=42
78   Planning Time: 1.891 ms
79   Execution Time: 95.237 ms

```

6. Времето изминато во извршување на операциите insert и update по индексирање изнесува:

```

[2026-09-10 14:47:43] vet_db.public> INSERT INTO treatment (examination_id, treatment_type_id, date_treatment, notes)
VALUES (1726, 2, CURRENT_DATE, 'Second follow-up dosage adjustment')
[2026-09-10 14:47:43] 1 row affected in 4 ms
[2026-09-10 14:48:12] vet_db.public> INSERT INTO treatment_attribute_value (treatment_id, treatment_attribute_id, value)
VALUES (577417, 13, 'false')
[2026-09-10 14:48:12] 1 row affected in 4 ms

[2026-09-10 14:52:20] vet_db.public> UPDATE treatment
SET notes = 'Updated dosage per second follow-up'
WHERE id = 1570
[2026-09-10 14:52:20] 1 row affected in 3 ms
[2026-09-10 14:52:20] vet_db.public> UPDATE treatment_attribute_value
SET value = 'true'
WHERE treatment_id = 1571 AND treatment_attribute_id = 13
[2026-09-10 14:52:20] 1 row affected in 3 ms

```

Забелешка: Индексот idx_treatment_attribute_value_treatment го отстрануваме затоа што и без него перформансите се добри. Времето на извршување на прашалникот е 702ms (92ms execution):

```

vet_db.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM pet_medical_history WHERE pet_id = 9956
60 rows retrieved starting from 1 in 702 ms (execution: 92 ms, fetching: 610 ms)

```

View3: Active prescriptions for pet

1. Примарен филтер за погледот active_prescriptions_for_pet ќе биде според id на милениче, а уште ќе се користи и според id на лек.
2. Примарен случај на употреба е пребарување кои лекови одредено милениче во моментот ги зема. Филтерот по medicine_id се користи кога клиниката треба да провери кои моментално активни пациенти се третираат со определен лек, на пример при повлекување на лек или проверка на интеракции меѓу лекови.
3. Иницијалното време на извршување на прашалникот по pet_id е 404ms (10ms execution) и 482ms (110ms execution) по medicine_id.

```
[2026-09-11 12:22:08] vet_db.public> SELECT * FROM active_prescriptions_for_pet WHERE pet_id = 3
[2026-09-11 12:22:09] 49 rows retrieved starting from 1 in 404 ms (execution: 10 ms, fetching: 394 ms)
[2026-09-11 12:22:12] vet_db.public> SELECT * FROM active_prescriptions_for_pet WHERE medicine_id = 47
[2026-09-11 12:22:12] 10 rows retrieved starting from 1 in 482 ms (execution: 110 ms, fetching: 372 ms)
```

4. Кај пребарувањето по pet_id ова е прифатливо време бидејќи сите операции во планот користат индекс и не постои непотребно скенирање на податоци, постоечкиот composite primary key на prescription_medicine и idx_appointment_pet веќе го покриваат овој пат на пребарување.

За пребарувањето по medicine_id табелата најбавната операција е seq scan на prescription за да се вратат само 10 совпаѓачки редици. Ова не е прифатливо па затоа пристапуваме кон индексирање.

10	Buffers: shared hit=1172
11	-> Nested Loop (cost=0.85..1834.02 rows=130 width=8) (actual time=0.026..0.416 rows=155.00 loops=1)
12	Buffers: shared hit=625
13	-> Index Scan using idx_appointment_pet on appointment a (cost=0.42..556.34 rows=154 width=8) (actual time=0.014..0.043 rows=155.00 loops=1)
14	Index Cond: (pet_id = 3)
15	Index Searches: 1
16	Buffers: shared hit=5
17	-> Index Scan using idx_examination_appointment on examination e (cost=0.42..8.29 rows=1 width=8) (actual time=0.002..0.002 rows=1.00 loops=155)
18	Index Cond: (appointment_id = a.id)
19	Index Searches: 155
20	Buffers: shared hit=620
21	-> Index Scan using prescription_examination_id_key on prescription pr (cost=0.42..0.48 rows=1 width=62) (actual time=0.002..0.002 rows=0.01 loops=155)
22	Index Cond: (examination_id = e.id)
23	Filter: ((date_start <= CURRENT_DATE) AND (date_end >= CURRENT_DATE))
24	Rows Removed by Filter: 1
25	Index Searches: 155
26	Buffers: shared hit=547
27	-> Index Scan using pet_pkey on pet p (cost=0.28..8.30 rows=1 width=14) (actual time=0.005..0.005 rows=1.00 loops=2)
28	Index Cond: (id = 3)
29	Index Searches: 2
30	Buffers: shared hit=6
31	-> Index Scan using owner_pkey on owner o (cost=0.28..8.29 rows=1 width=17) (actual time=0.004..0.004 rows=1.00 loops=2)
32	Index Cond: (id = p.owner_id)
33	Index Searches: 2
34	Buffers: shared hit=6
35	-> Index Scan using prescription_medicine_pkey on prescription_medicine pm (cost=0.43..28.56 rows=47 width=16) (actual time=0.007..0.011 rows=24.50 loops=2)
36	Index Cond: (prescription_id = pr.id)
37	Index Searches: 2
38	Buffers: shared hit=8

5	QUERY PLAN
6	-> Nested Loop (cost=2.13..24530.92 rows=7 width=147) (actual time=3.794..28.720 rows=3.33 loops=3)
7	Buffers: shared hit=12788 read=4157
8	-> Nested Loop (cost=1.84..24528.50 rows=7 width=97) (actual time=3.765..28.688 rows=3.33 loops=3)
9	Buffers: shared hit=12777 read=4157
10	-> Nested Loop (cost=1.57..24526.19 rows=7 width=84) (actual time=3.749..28.659 rows=3.33 loops=3)
11	Buffers: shared hit=12745 read=4157
12	-> Nested Loop (cost=1.28..24524.08 rows=7 width=74) (actual time=3.731..28.625 rows=3.33 loops=3)
13	Buffers: shared hit=12713 read=4157
14	-> Nested Loop (cost=0.86..24520.53 rows=7 width=74) (actual time=3.711..28.588 rows=3.33 loops=3)
15	Buffers: shared hit=12671 read=4157
16	-> Nested Loop (cost=0.43..24482.70 rows=7 width=74) (actual time=3.688..28.547 rows=3.33 loops=3)
17	Buffers: shared hit=12629 read=4157
18	-> Parallel Seq Scan on prescription pr (cost=0.00..8624.65 rows=1982 width=62) (actual time=0.098..23.562 rows=1305.67 loops=3)
19	Filter: ((date_start <= CURRENT_DATE) AND (date_end >= CURRENT_DATE))
20	Rows Removed by Filter: 142760
21	Buffers: shared hit=866 read=4157
22	-> Index Scan using prescription_medicine_pkey on prescription_medicine pm (cost=0.43..8.00 rows=1 width=16) (actual time=0.004..0.004 rows=1 loops=3)
23	Index Cond: ((prescription_id = pr.id) AND (medicine_id = 47))
24	Index Searches: 3917
25	Buffers: shared hit=11763

Времето изминато во извршување на операциите insert и update изнесува:

```
[2026-09-11 12:41:36] vet_db.public> INSERT INTO prescription (examination_id, date_start, date_end, description)
VALUES (699, CURRENT_DATE, CURRENT_DATE + 10, 'Antibiotic course, 10 days')
[2026-09-11 12:41:36] 1 row affected in 15 ms
[2026-09-11 12:42:02] vet_db.public> INSERT INTO prescription_medicine (prescription_id, medicine_id, dosage, num_days)
VALUES (432525, 1, 250, 10)
[2026-09-11 12:42:02] 1 row affected in 10 ms

[2026-09-11 12:44:56] vet_db.public> UPDATE prescription
SET date_end = date_end + 5
WHERE id = 328
[2026-09-11 12:44:56] 1 row affected in 6 ms
[2026-09-11 12:44:56] vet_db.public> UPDATE prescription_medicine
SET dosage = 300
WHERE prescription_id = 327 AND medicine_id = 8
[2026-09-11 12:44:56] 1 row affected in 4 ms

CREATE INDEX idx_prescription_active_dates ON prescription (date_end, date_start);
```

- Времето изминато во извршување на query-то со индекси изнесува 346ms (7ms execution) по pet_id и 355ms (21ms execution) по medicine_id и тоа е прифатливо време.

```
[2026-09-11 12:55:32] vet_db.public> SELECT * FROM active_prescriptions_for_pet WHERE pet_id = 3
[2026-09-11 12:55:32] 49 rows retrieved starting from 1 in 346 ms (execution: 7 ms, fetching: 339 ms)
[2026-09-11 12:55:35] vet_db.public> SELECT * FROM active_prescriptions_for_pet WHERE medicine_id = 47
[2026-09-11 12:55:36] 10 rows retrieved starting from 1 in 355 ms (execution: 21 ms, fetching: 334 ms)
```

QUERY PLAN	
7	-> Nested Loop (cost=131.45..20185.47 rows=7 width=97) (actual time=0.495..5.799 rows=3.33 loops=3)
8	Buffers: shared hit=14717
9	-> Nested Loop (cost=131.17..20183.16 rows=7 width=84) (actual time=0.493..5.777 rows=3.33 loops=3)
10	Buffers: shared hit=14687
11	-> Nested Loop (cost=130.89..20181.06 rows=7 width=74) (actual time=0.490..5.758 rows=3.33 loops=3)
12	Buffers: shared hit=14657
13	-> Nested Loop (cost=130.46..20177.51 rows=7 width=74) (actual time=0.487..5.735 rows=3.33 loops=3)
14	Buffers: shared hit=14617
15	-> Nested Loop (cost=130.04..20138.60 rows=7 width=74) (actual time=0.483..5.709 rows=3.33 loops=3)
16	Buffers: shared hit=14577
17	-> Parallel Bitmap Heap Scan on prescription pr (cost=129.60..5253.13 rows=1854 width=62) (actual time=0.374..1.458 rows=1306.00 loops=3)
18	Recheck Cond: ((date_end >= CURRENT_DATE) AND (date_start <= CURRENT_DATE))
19	Heap Blocks: exact=2765
20	Buffers: shared hit=2813
21	-> Bitmap Index Scan on idx_prescription_active_dates (cost=0.00..128.49 rows=4449 width=0) (actual time=0.513..0.513 rows=3918.00 loops=1)
22	Index Cond: ((date_end >= CURRENT_DATE) AND (date_start <= CURRENT_DATE))
23	Index Searches: 1
24	Buffers: shared hit=8
25	-> Index Scan using prescription_medicine_pkey on prescription_medicine pm (cost=0.43..8.03 rows=1 width=16) (actual time=0.003..0.003 rows=0.00 loops=1)
26	Index Cond: ((prescription_id = pr.id) AND (medicine_id = 47))
27	Index Searches: 3918
28	Buffers: shared hit=11764

6. Времето изминато во извршување на операциите insert и update по индексирање изнесува:

```
[2026-09-11 13:02:48] vet_db.public> INSERT INTO prescription (examination_id, date_start, date_end, description)
VALUES (700, CURRENT_DATE, CURRENT_DATE + 7, 'Follow-up course')
[2026-09-11 13:02:48] 1 row affected in 3 ms
[2026-09-11 13:02:59] vet_db.public> INSERT INTO prescription_medicine (prescription_id, medicine_id, dosage, num_days)
VALUES (432526, 2, 200, 7)
[2026-09-11 13:02:59] 1 row affected in 2 ms
```

```
[2026-09-11 13:05:05] vet_db.public> UPDATE prescription
SET date_end = date_end + 3
WHERE id = 329
[2026-09-11 13:05:05] 1 row affected in 3 ms
[2026-09-11 13:05:06] vet_db.public> UPDATE prescription_medicine
SET dosage = 150
WHERE prescription_id = 329 AND medicine_id = 6756
[2026-09-11 13:05:06] 1 row affected in 2 ms
```

Забелешка: Индексот idx_prescription_active_dates го отстрануваме. Филтрирањето по датум се прави по спојувањето на appointment, examination, prescription кои веќе користат индекси по примарните клучеви.

View4: Employee Performance Summary

1. Примарен филтер за пребарување кај погледот `employee_performance_summary` е според `id` на вработен во клиниката, но исто така може да се пребарува и преку датум на извршен третман или пак одреден период помеѓу два датуми ако набљудувачот сака подетални информации.

2. Примарен случај за користење на овој поглед е при извршување на евалуација на вработените. При пребарување по `id` на вработен за него се прикажува неговата улога во клиниката, сите прегледи кои ги има извршено како и фактурите кои припаѓаат на тие прегледи, со цел детална евалуација на работната на вработениот и успешно извршување на задачите на клиниката. Пребарувањето по датум служи за периодични евалуации на вработените преку кои може да се погледнат направените прегледи во даден временски период или на одреден датум.

3. Иницијалното време на извршување на погледот при пребарување по `employee_id` е: 1 in 1 s 119 ms (execution: 537 ms, fetching: 582 ms). За пребарување по `date_examination` е: 1 in 884 ms (execution: 248 ms, fetching: 636 ms).

```
[2026-09-15 21:55:57] vetDB.public> SELECT * FROM employee_performance_summary WHERE employee_id = 5
[2026-09-15 21:55:58] 500 rows retrieved starting from 1 in 1 s 119 ms (execution: 537 ms, fetching: 582 ms)
[2026-09-15 21:55:58] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM employee_performance_summary WHERE employee_id = 5
[2026-09-15 21:56:00] 50 rows retrieved starting from 1 in 1 s 859 ms (execution: 1 s 498 ms, fetching: 361 ms)
[2026-09-15 22:06:15] vetDB.public> SELECT * FROM employee_performance_summary
    WHERE employee_id = 5 AND date_treatment >= '2025-01-01' AND date_treatment < '2026-01-01'
[2026-09-15 22:06:16] 500 rows retrieved starting from 1 in 884 ms (execution: 248 ms, fetching: 636 ms)
[2026-09-15 22:06:16] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM employee_performance_summary
    WHERE employee_id = 5 AND date_treatment >= '2025-01-01' AND date_treatment < '2026-01-01'
[2026-09-15 22:06:18] 52 rows retrieved starting from 1 in 1 s 848 ms (execution: 1 s 128 ms, fetching: 720 ms)
```

4. За разлика од претходните погледи, ова е **MATERIALIZED VIEW**. Резултатот е физички зачуван, па `SELECT`-от го чита готовиот резултат наместо секој пат да го пресметува `join`-от. Со материјализирање на погледот, тешкиот `join` (кон `invoice_item`, `treatment`, `examination`) се пресметува еднаш при `REFRESH`, наместо при секој `SELECT`. Ова значи дека не мора да се додаваат дополнителни индекси на базните трансакциски табели (пред сè `invoice_item`, која е голема и често се ажурира) само за да се забрза овој поглед.

Баш заради ова беа отстранети некои од индексите кои ги испробавме, затоа што не придонесуваа значително подобрување на перформансите на погледот.

Најбавната операција е `Parallel Seq Scan on invoice_item` — целата табела `invoice_item` (околу 1.9 милиони редици, ~34 673 блока прочитани од диск по `worker`) се скенира

секвенцијално за да се направи Hash Right Join по `ii.treatment_id = t.id`, што трошинајголем дел од вкупното време (~880ms). Дополнително придонесува и Parallel Seq Scan on treatment (239 938 редици, 7234 блока read, ~58ms по worker).

The screenshot shows a query plan for a join operation. The plan is displayed in a table format with the following details:

Step	Operation	Details
34	Buffers: shared hit=3	
35	Buffers: shared hit=11283	
36	Buckets: 131072 Batches: 1 Memory Usage: 5440...	
37	-> Parallel Seq Scan on invoice_item ii (cost...	
38	-> Parallel Hash (cost=10871.07..10871.07 row...	
39	-> Parallel Bitmap Heap Scan on examination e ...	
40	Recheck Cond: (employee_id = 5)	
41	Heap Blocks: exact=9963	
42	Buffers: shared hit=3632 read=34545	
43	Buffers: shared hit=166 read=7106	
44	Buffers: shared hit=11283	
45	Buckets: 131072 Batches: 1 Memory Usage...	
46	-> Parallel Seq Scan on treatment t (co...	
47	-> Bitmap Index Scan on idx_examination_...	
48	Rows Removed by Filter: 153453	
49	Index Cond: (employee_id = 5)	
50	Filter: ((date_treatment >= '2025-...	
51	Buffers: shared hit=71	

Време на избршување на INSERT и UPDATE операции пред индексирање:

```
[2026-09-15 22:21:44] vetDB.public> INSERT INTO examination(date_examination, status, description, appointment_id, employee_id, examinat
VALUES ('2026-08-01','completed','Regular check-up', 720800,6,3)
[2026-09-15 22:21:44] 1 row affected in 95 ms
[2026-09-15 22:21:44] vetDB.public> INSERT INTO treatment(date_treatment, notes, treatment_type_id, examination_id)
VALUES ('2026-08-01','Follow-up plan agreed',2,715)
[2026-09-15 22:21:44] 1 row affected in 25 ms
[2026-09-15 22:21:44] vetDB.public> INSERT INTO invoice(date_invoice, total, coupon_id, owner_id)
VALUES ('2026-08-01',56.33,398,79)
[2026-09-15 22:21:44] 1 row affected in 20 ms
[2026-09-15 22:21:44] vetDB.public> INSERT INTO invoice_item(num_item, invoice_id, price,quantity, type, shop_item_id, treatment_id)
VALUES (1,24972,67.87,1,'treatment',null,223433)
[2026-09-15 22:21:47] 1 row affected in 2 s 551 ms
[2026-09-15 22:21:47] vetDB.public> UPDATE treatment
SET notes = 'Updated: discussed follow-up diet plan'
WHERE id = 5778001
[2026-09-15 22:21:47] completed in 11 ms
```

5. За оптимизација е креирани следниот индекс:

```
CREATE INDEX idx_treatment_examination ON treatment (examination_id);
```

Време на извршување на пребарување по индексирање

```

[2026-09-15 22:23:01] vetDB.public> SELECT * FROM employee_performance_summary WHERE employee_id = 5
[2026-09-15 22:23:02] 500 rows retrieved starting from 1 in 837 ms (execution: 388 ms, fetching: 449 ms)
[2026-09-15 22:23:02] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM employee_performance_summary WHERE employee_id = 5
[2026-09-15 22:23:03] 50 rows retrieved starting from 1 in 1 s 161 ms (execution: 791 ms, fetching: 370 ms)
[2026-09-15 22:23:03] vetDB.public> SELECT * FROM employee_performance_summary
    WHERE employee_id = 5 AND date_treatment >= '2025-01-01' AND date_treatment < '2026-01-01'
[2026-09-15 22:23:04] 500 rows retrieved starting from 1 in 783 ms (execution: 251 ms, fetching: 532 ms)
[2026-09-15 22:23:04] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM employee_performance_summary
    WHERE employee_id = 5 AND date_treatment >= '2025-01-01' AND date_treatment < '2026-01-01'
[2026-09-15 22:23:05] 52 rows retrieved starting from 1 in 1 s 202 ms (execution: 827 ms, fetching: 375 ms)

```

6. Време за извршување INSERT и UPDATE операции по индексирање:

```

[2026-09-15 22:24:17] vetDB.public> INSERT INTO examination(date_examination, status, description, appointment_id, employee_id, examination_id)
    VALUES ('2026-08-01','completed','Regular check-up', 720800,6,3)
[2026-09-15 22:24:17] 1 row affected in 42 ms
[2026-09-15 22:24:17] vetDB.public> INSERT INTO treatment(date_treatment, notes, treatment_type_id, examination_id)
    VALUES ('2026-08-01','Follow-up plan agreed',2,715)
[2026-09-15 22:24:17] 1 row affected in 7 ms
[2026-09-15 22:24:17] vetDB.public> INSERT INTO invoice(date_invoice, total, coupon_id, owner_id)
    VALUES ('2026-08-01',56.33,398,79)
[2026-09-15 22:24:17] 1 row affected in 8 ms
[2026-09-15 22:24:17] vetDB.public> INSERT INTO invoice_item(num_item, invoice_id, price,quantity, type, shop_item_id, treatment_id)
    VALUES (1,24972,67.87,1,'treatment',null,223433)
[2026-09-15 22:24:17] 1 row affected in 137 ms
[2026-09-15 22:24:17] vetDB.public> UPDATE treatment
    SET notes = 'Updated: discussed follow-up diet plan'
    WHERE id = 5778001
[2026-09-15 22:24:17] completed in 5 ms

```

View5: Invoice Billing Details

1. Примарен филтер за погледот invoice_billing_details ќе биде според invoice_id, но може да се пребарува и по owner_id
2. Примарен случај на употреба на овој поглед е вработените кои работат на каса да можат да ги разгледаат сите детали за една фактурна ставка. Значи да имаат преглед за сопственикот, третманите и производите кои сопственикот ги купува за своето милениче како и купони кои биле искористени во текот на наплатата.
3. Иницијално време на извршување на погледот при пребарување по invoice_id е: 1 in 1 s 383 ms (execution: 963 ms, fetching: 420 ms), а при пребарување по owner_id е: 1 in 1 s 721 ms (execution: 1 s 200 ms, fetching: 521 ms)

```
[2026-09-15 22:30:41] vetDB.public> SELECT * FROM invoice_billing_details WHERE invoice_id = 500000
[2026-09-15 22:30:43] 1 row retrieved starting from 1 in 1 s 383 ms (execution: 963 ms, fetching: 420 ms)
[2026-09-15 22:30:43] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM invoice_billing_details WHERE invoice_id = 500000
[2026-09-15 22:30:44] 54 rows retrieved starting from 1 in 817 ms (execution: 402 ms, fetching: 415 ms)
[2026-09-15 22:30:44] vetDB.public> SELECT * FROM invoice_billing_details WHERE owner_id = 45
[2026-09-15 22:30:45] 500 rows retrieved starting from 1 in 1 s 721 ms (execution: 1 s 200 ms, fetching: 521 ms)
[2026-09-15 22:30:45] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM invoice_billing_details WHERE owner_id = 45
[2026-09-15 22:30:47] 62 rows retrieved starting from 1 in 1 s 278 ms (execution: 899 ms, fetching: 379 ms)
```

4. Кај пребарувањето по invoice id времето на извршување е 392.264 ms.

Пребарувањето на `invoice`, `owner` и `coupon` користи постоечки `primary key` индекси, но најбавните операции се `Parallel Seq Scan` над табелите `payment` и `invoice_item`. Особено кај `invoice_item`, се скенираат голем број записи иако се бараат само записите за конкретен `invoice_id`. Затоа е потребно индексирање на `invoice_item(invoice_id, num_item)`, додека за `payment` е предвиден индексот `idx_payment_invoice`.

Кај пребарувањето по `owner_id` времето на извршување е 889.622 ms, што не е прифатливо за пребарување по конкретен `owner`. Најбавните операции се `Parallel Seq Scan` над `invoice` и `invoice_item`. Табелата `invoice` се скенира за да се пронајдат фактурите со `owner_id = 45`, додека `invoice_item` се скенира за да се пронајдат ставките што припаѓаат на тие фактури. Затоа се користат индексите `idx_invoice_owner` и `idx_invoice_item_invoice_id`, со што се избегнува непотребно скенирање на целите табели.

```
Services
Tx, +, >
Output Result 40-2 vetDB.public.invoice_billing_details vetDB.public.invoice_billing_details 2 Result 40-4 x
QUERY PLAN
27 Hash Cond: (i.coupon_id = c.id)
28 Buffers: shared hit=255 read=47863
29 -> Parallel Hash Right Join (cost=14968.98..77209.54 rows=908 width=54) (actual time=78.456...)
30 Hash Cond: (ii.invoice_id = i.id)
31 Buffers: shared hit=198 read=47863
32 -> Parallel Seq Scan on invoice_item ii (cost=0.00..57237.23 rows=1906023 width=36) (a...
33 Buffers: shared hit=163 read=38014
34 -> Parallel Hash (cost=14966.55..14966.55 rows=194 width=22) (actual time=75.091..75.0...)
35 Buckets: 1024 Batches: 1 Memory Usage: 104kB
36 Buffers: shared hit=35 read=9849
37 -> Parallel Seq Scan on invoice i (cost=0.00..14966.55 rows=194 width=22) (actua...
38 Filter: (owner_id = 45)
39 Rows Removed by Filter: 325172
40 Buffers: shared hit=35 read=9849
41 -> Hash (cost=29.00..29.00 rows=1000 width=26) (actual time=1.012..1.013 rows=1000 loops=3)
42 Buckets: 1024 Batches: 1 Memory Usage: 69kB
43 Buffers: shared hit=57
44 -> Seq Scan on coupon c (cost=0.00..29.00 rows=1000 width=26) (actual time=0.573..0.77...)
45 Buffers: 62 rows : 7
```

Services	
Output	Result 42-2 × vetDB.public.invoice_billing_details vetDB.public.invoice_billing_details 2 Result 40-4
vet	QUERY PLAN
18	Workers Planned: 2
19	Workers Launched: 2
20	Buffers: shared hit=321 read=5282
21	-> Parallel Seq Scan on payment p (cost=0.00..9643.88 rows=1 width=23) (actual time=65.058..65.083 rows=0 loops=3)
22	Filter: (invoice_id = 500000)
23	Rows Removed by Filter: 258617
24	Buffers: shared hit=321 read=5282
25	-> Gather (cost=1008.18..63045.33 rows=10 width=582) (actual time=69.872..383.083 rows=1 loops=1)
26	Workers Planned: 2
27	Workers Launched: 2
28	Buffers: shared hit=300 read=37886
29	-> Nested Loop Left Join (cost=8.18..62044.33 rows=4 width=582) (actual time=188.451..287.314 rows=0 loops=3)
30	Buffers: shared hit=300 read=37886
31	-> Nested Loop Left Join (cost=8.04..62043.68 rows=4 width=70) (actual time=188.443..287.306 rows=0 loops=3)
32	Buffers: shared hit=298 read=37886
33	-> Hash Left Join (cost=7.61..62009.91 rows=4 width=66) (actual time=188.431..287.294 rows=0 loops=3)
34	Hash Cond: (ii.shop_item_id = si.id)
35	Buffers: shared hit=294 read=37886
36	-> Parallel Seq Scan on invoice_item ii (cost=0.00..62007.29 rows=4 width=36) (actual time=188.374..287.23..
37	Filter: (invoice_id = 500000) 54 rows v :

Време на избршување на INSERT и UPDATE операции пред индексирање:

```
[2026-09-15 22:44:02] vetDB.public> INSERT INTO invoice (date_invoice, total, coupon_id, owner_id)
VALUES (CURRENT_DATE, 62.50, NULL, 8)
[2026-09-15 22:44:02] 1 row affected in 4 ms
[2026-09-15 22:44:02] vetDB.public> INSERT INTO invoice_item (num_item, invoice_id, price, quantity, type, shop_item_id, treatment_id)
VALUES (1, 24984, 62.50, 1, 'shop_item', 9, null)
[2026-09-15 22:44:03] 1 row affected in 933 ms
[2026-09-15 22:44:03] vetDB.public> INSERT INTO payment (date_payment, amount, method, invoice_id)
VALUES (CURRENT_DATE, 62.50, 'credit card', 24984)
[2026-09-15 22:44:03] 1 row affected in 38 ms
[2026-09-15 22:44:03] vetDB.public> UPDATE invoice
SET total = 58.00
WHERE id = 24984
[2026-09-15 22:44:03] 1 row affected in 10 ms
[2026-09-15 22:44:03] vetDB.public> UPDATE payment
SET amount = 58.00
WHERE invoice_id = 24984
[2026-09-15 22:44:03] 4 rows affected in 155 ms
```

5. За оптимизација се креирани следните индекси:

CREATE INDEX idx_invoice_owner ON invoice (owner_id);

CREATE INDEX idx_payment_invoice ON payment (invoice_id);

Време потребно за извршување на SELECT операции по индексирање

```

[2026-09-15 22:48:13] vetDB.public> SELECT * FROM invoice_billing_details WHERE invoice_id = 500000
[2026-09-15 22:48:14] 1 row retrieved starting from 1 in 951 ms (execution: 566 ms, fetching: 385 ms)
[2026-09-15 22:48:14] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM invoice_billing_details WHERE invoice_id = 500000
[2026-09-15 22:48:15] 49 rows retrieved starting from 1 in 694 ms (execution: 356 ms, fetching: 338 ms)
[2026-09-15 22:48:15] vetDB.public> SELECT * FROM invoice_billing_details WHERE owner_id = 45
[2026-09-15 22:48:16] 500 rows retrieved starting from 1 in 854 ms (execution: 410 ms, fetching: 444 ms)
[2026-09-15 22:48:16] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM invoice_billing_details WHERE owner_id = 45
[2026-09-15 22:48:17] 66 rows retrieved starting from 1 in 1 s 152 ms (execution: 811 ms, fetching: 341 ms)

```

6. Време за извршување INSERT и UPDATE операции по индексирање:

```

[2026-09-15 23:22:59] vetDB.public> INSERT INTO invoice (date_invoice, total, coupon_id, owner_id)
VALUES (CURRENT_DATE, 62.50, NULL, 8)
[2026-09-15 23:22:59] 1 row affected in 15 ms
[2026-09-15 23:22:59] vetDB.public> INSERT INTO invoice_item (num_item, invoice_id, price, quantity, type, shop_item_id, treatment_id)
VALUES (1, 24984, 62.50, 1, 'shop_item', 9, null)
[2026-09-15 23:22:59] 1 row affected in 171 ms
[2026-09-15 23:22:59] vetDB.public> INSERT INTO payment (date_payment, amount, method, invoice_id)
VALUES (CURRENT_DATE, 62.50, 'credit card', 24984)
[2026-09-15 23:22:59] 1 row affected in 13 ms
[2026-09-15 23:22:59] vetDB.public> UPDATE invoice
SET total = 58.00
WHERE id = 24984
[2026-09-15 23:22:59] 1 row affected in 11 ms
[2026-09-15 23:22:59] vetDB.public> UPDATE payment
SET amount = 58.00
WHERE invoice_id = 24984
[2026-09-15 23:22:59] 5 rows affected in 6 ms

```

View6: Low Stock Shop Items

1. Primary Filter: Основното пребарување е според името на категоријата на продукти category_name (но исто функционира и со ниво на залиха)
2. Primary Case: Погледот low_stock_shop_items ги прикажува производите со помалку од 20 единици на залиха, заедно со нивната цена, категорија и наредена категорија. Се користи за полесно идентификување на производите кои треба да се надополнат. Производите се поврзуваат со нивните категории, додека LEFT JOIN овозможува прикажување на наредената категорија доколку постои. Погледот поддржува пребарување според категорија и дополнително филтрирање според ниво на залиха.
3. Иницијално време на извршување на погледот при пребарување по category_name е: 1 in 474 ms (execution: 25 ms, fetching: 449 ms) , а при пребарување по stock treshold е: 11 in 449 ms (execution: 9 ms, fetching: 440 ms)

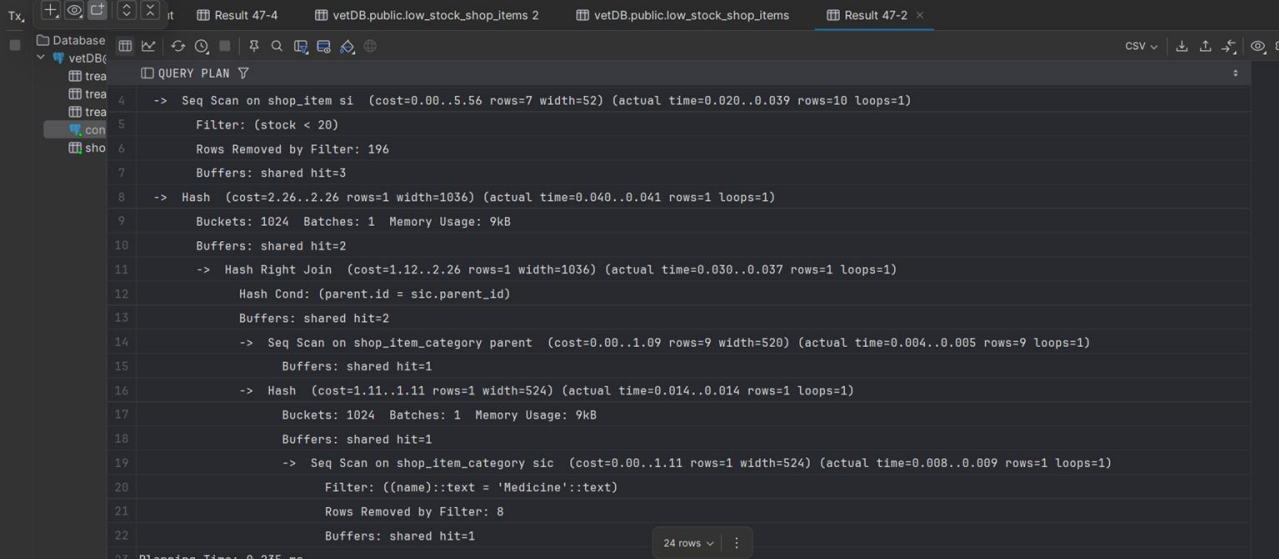
```

[2026-09-15 23:39:17] vetDB.public> SELECT * FROM low_stock_shop_items WHERE category_name = 'Medicine'
[2026-09-15 23:39:17] 7 rows retrieved starting from 1 in 474 ms (execution: 25 ms, fetching: 449 ms)
[2026-09-15 23:39:17] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM low_stock_shop_items WHERE category_name = 'Medicine'
[2026-09-15 23:39:18] 24 rows retrieved starting from 1 in 439 ms (execution: 9 ms, fetching: 430 ms)
[2026-09-15 23:39:18] vetDB.public> SELECT * FROM low_stock_shop_items WHERE stock < 10
[2026-09-15 23:39:18] 4 rows retrieved starting from 1 in 449 ms (execution: 9 ms, fetching: 440 ms)
[2026-09-15 23:39:18] vetDB.public> EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM low_stock_shop_items WHERE stock < 10
[2026-09-15 23:39:19] 18 rows retrieved starting from 1 in 492 ms (execution: 9 ms, fetching: 483 ms)

```

4. Кај пребарувањето по `stock < 10`, времето на извршување е само 0.103 ms, што е прифатливо. Планот користи Seq Scan над `shop_item`, при што се проверуваат сите записи и се враќаат само оние кои го задоволуваат условот `stock < 10`. Бидејќи табелата содржи мал број записи, целосното скенирање не претставува проблем и додавање индекс за овој услов не е потребно. Во планот се појавуваат условите `stock < 20` и `stock < 10`, бидејќи првиот услов е дефиниран во погледот, а вториот е дополнителен услов во пребарувањето.

Кај пребарувањето по `category_name = 'Medicine'`, времето на извршување е 0.137 ms, што исто така е прифатливо. Планот користи Seq Scan над `shop_item` и `shop_item_category`, како и Hash Join за нивно поврзување. Бидејќи `shop_item` и `shop_item_category` се мали табели, скенирањето на сите записи не создава значително оптоварување. Поради тоа, не е потребно дополнително индексирање за овие пребарувања.



Step	Operation	Cost	Rows	Width	Actual Time	Loops
4	Seq Scan on shop_item si	(cost=0.00..5.56 rows=7 width=52)	7	52	(actual time=0.020..0.039 rows=10)	1
5	Filter: (stock < 20)					
6	Rows Removed by Filter: 196					
7	Buffers: shared hit=3					
8	Hash	(cost=2.26..2.26 rows=1 width=1036)	1	1036	(actual time=0.040..0.041 rows=1)	1
9	Buckets: 1024 Batches: 1 Memory Usage: 9kB					
10	Buffers: shared hit=2					
11	Hash Right Join	(cost=1.12..2.26 rows=1 width=1036)	1	1036	(actual time=0.030..0.037 rows=1)	1
12	Hash Cond: (parent.id = sic.parent_id)					
13	Buffers: shared hit=2					
14	Seq Scan on shop_item_category parent	(cost=0.00..1.09 rows=9 width=520)	9	520	(actual time=0.004..0.005 rows=9)	1
15	Buffers: shared hit=1					
16	Hash	(cost=1.11..1.11 rows=1 width=524)	1	524	(actual time=0.014..0.014 rows=1)	1
17	Buckets: 1024 Batches: 1 Memory Usage: 9kB					
18	Buffers: shared hit=1					
19	Seq Scan on shop_item_category sic	(cost=0.00..1.11 rows=1 width=524)	1	524	(actual time=0.008..0.009 rows=1)	1
20	Filter: ((name)::text = 'Medicine'::text)					
21	Rows Removed by Filter: 8					
22	Buffers: shared hit=1					
23	Planning Time: 0.235 ms					

```

QUERY PLAN
Nested Loop Left Join (cost=0.00..8.48 rows=1 width=1084) (actual time=0.034..0.079 rows=4 loops=1)
  Join Filter: (parent.id = sic.parent_id)
  Rows Removed by Join Filter: 5
  Buffers: shared hit=11
  -> Nested Loop (cost=0.00..7.28 rows=1 width=572) (actual time=0.029..0.062 rows=4 loops=1)
    Join Filter: (sic.id = si.shop_item_category_id)
    Rows Removed by Join Filter: 10
    Buffers: shared hit=7
    -> Seq Scan on shop_item si (cost=0.00..6.07 rows=1 width=52) (actual time=0.021..0.040 rows=4 loops=1)
      Filter: ((stock < 20) AND (stock < 10))
      Rows Removed by Filter: 202
      Buffers: shared hit=3
    -> Seq Scan on shop_item_category sic (cost=0.00..1.09 rows=9 width=524) (actual time=0.002..0.002 rows=4 loops=4)
      Buffers: shared hit=4
  -> Seq Scan on shop_item_category parent (cost=0.00..1.09 rows=9 width=520) (actual time=0.001..0.001 rows=2 loops=4)
    Buffers: shared hit=4
Planning Time: 0.242 ms
Execution Time: 0.103 ms

```

Извршување на INSERT и UPDATE операции

```

[2026-09-15 23:46:44] vetDB.public> INSERT INTO shop_item (name, price, stock, shop_item_category_id)
                                SELECT 'Emergency Saline Restock Pack', 12.99, 8, sic.id
                                FROM shop_item_category sic
                                WHERE sic.name = 'Medicine'
                                LIMIT 1
[2026-09-15 23:46:44] 1 row affected in 7 ms
[2026-09-15 23:46:44] vetDB.public> UPDATE shop_item
                                SET stock = stock - 5
                                WHERE id = (
                                    SELECT id
                                    FROM shop_item
                                    WHERE name = 'Emergency Saline Restock Pack'
                                    ORDER BY id DESC
                                    LIMIT 1
                                )
                                AND stock >= 5
[2026-09-15 23:46:44] 1 row affected in 4 ms
[2026-09-15 23:46:44] vetDB.public> UPDATE shop_item
                                SET stock = stock - 3
                                WHERE id = (
                                    SELECT id
                                    FROM shop_item
                                    WHERE name = 'Emergency Saline Restock Pack'
                                    ORDER BY id DESC
                                    LIMIT 1
                                )
                                AND stock >= 3

```