

Напредни бази на податоци Фаза 4 - Индекси и оптимизација на прашалници

Проект: Paws and Pillows
Ана Мацановиќ 231178
Антонио Крпачовски 231024
Лина Ангелковска 231123

View1: My bookings

- Примарен филтер за погледот vw_my_bookings ќе биде според id на корисникот, а може да се користи и дополнитено со име на миленичето.
- Примарен случај на употреба ќе е преглед на сите резервации кои ги има направено корисник. За овој поглед ни се важни перформансите, бидејќи ќе биде користен секојдневно од клиенти. Иницијалното време за извршување на погледот е 5s 356ms.

```
[2026-07-01 16:21:14] postgres.public> SELECT * FROM vw_my_bookings
                                WHERE customer_id = 100286
                                ORDER BY reservation_date DESC
[2026-07-01 16:21:20] 500 rows retrieved starting from 1 in 5 s 356 ms (execution: 4 s 952 ms, fetching: 404 ms)
```

- Ова не е прифатливо време за апликацијата и query-то не може понатаму да се оптимизира со менување на логиката, па затоа пристапуваме кон индексирање.

Plan Step	Table/Index	Rows	Time (ms)	Cost
Full Scan (Seq Scan)	reservation;	4166667	135416.67	0
Transformation (Hash)		1	1956.12	1956.12
Full Scan (Seq Scan)	table: pet;	1	1956.12	0
Index Scan	table: customer; index: customer_pkey;	1	8.44	0.42
Unknown (Memoize)		1	0.18	0.16
Index Scan	table: reservationstatus; index: reservationstatus_pkey;	1	0.17	0.15
Index Scan	table: roomreservation; index: uq_roomreservation_reservation;	1	0.49	0.43
Index Scan	table: room; index: room_pkey;	1	0.3	0.28
Unknown (Memoize)		1	0.19	0.15
Index Scan	table: room_type; index: room_type_pkey;	1	0.18	0.14
Unknown (Memoize)		1	0.19	0.15
Index Scan	table: hotel; index: hotel_pkey;	1	0.18	0.14
Transformation (Hash)		3810250	105342.5	105342.5
Full Scan (Seq Scan)	table: servicereservation;	3810250	105342.5	0
Unknown (Memoize)		1	0.18	0.16
Index Scan	table: servicereservationstatus; index: servicereservationstatus_pkey;	1	0.17	0.15
Transformation (Hash)		2777875	70243.75	70243.75
Full Scan (Seq Scan)	table: payment;	2777875	70243.75	0
Unknown (Memoize)		1	0.17	0.15
Index Scan	table: service; index: service_pkey;	1	0.16	0.14
Unknown (Memoize)		1	0.18	0.16
Index Scan	table: paymentmethod; index: paymentmethod_pkey;	1	0.17	0.15
Unknown (Memoize)		1	0.18	0.16
Index Scan	table: paymentstatus; index: paymentstatus_pkey;	1	0.17	0.15

```
CREATE INDEX idx_payment_reservation_id ON Payment (reservation_id);
CREATE INDEX idx_petID_reservation ON reservation(pet_id);
```

- Се приметува дека главниот проблем е Full Scan на табелата reservation и payment, и бидејќи и правиме join преку pet_id на reservation и join преку reservation_id на payment, можеме да ја индексираме по тие податоци.

- Со помош на овие индекси времето за извршување значително се намалува на 2s 665ms

```
[2026-07-01 17:07:13] postgres.public> SELECT * FROM vw_my_bookings
      WHERE customer_id = 100286
      ORDER BY reservation_date DESC
[2026-07-01 17:07:16] 500 rows retrieved starting from 1 in 2 s 665 ms (execution: 2 s 283 ms, fetching: 382 ms)
```

▼ Nested Loops (Nest)		47	3510.19	7.97
▼ Nested Loops (Nest)		1	1964.57	0.42
Full Scan (Seq Scan) on;		1	1956.12	0
Index Scan	table: customer; index: customer_pkey;	1	8.44	0.42
▼ Bitmap Index Scan	table: reservation;	402	1541.6	7.55
Index Scan	index: idx_petid_reservation;	402	7.45	0
Unknown (Memoize)		1	0.18	0.16
Index Scan	table: reservationstatus; index: reservationstatus_pkey;	1	0.17	0.15
Index Scan	table: roomreservation; index: uq_roomreservation_reservation;	1	0.49	0.43
Index Scan	table: room; index: room_pkey;	1	0.3	0.28
Unknown (Memoize)		1	0.19	0.15
Index Scan	table: room_type; index: room_type_pkey;	1	0.18	0.14
Unknown (Memoize)		1	0.19	0.15
Index Scan	table: hotel; index: hotel_pkey;	1	0.18	0.14
Transformation (Hash)		3810250	105342.5	105342.5
Full Scan (Seq Scan)	table: servicesreservation;	3810250	105342.5	0
Transformation (Hash)		2777875	70243.75	70243.75

View2: pending actions

- Примарен филтер за погледот vw_my_pending_actions ќе биде според id на корисникот.
- Примарен случај на употреба ќе е преглед на сите идни резервации кои сеуште ги нема потврдено ("pending") или кои сеуште ги нема платено. За овој поглед ни се важни перформансите, бидејќи ќе биде користен секојдневно од клиенти. Иницијалното време за извршување на погледот е 6s 949ms.

```
[2026-07-01 16:35:16] postgres.public> select * from vw_my_pending_actions
      where customer_id = 100286
[2026-07-01 16:35:23] 100 rows retrieved starting from 1 in 6 s 949 ms (execution: 6 s 617 ms, fetching: 332 ms)
```

- Ова не е прифатливо време за апликацијата и query-то не може понатаму да се оптимизира со менување на логиката, па затоа пристапуваме кон индексирање.

Plan	Table/Index	Rows	Cost	Time
Index Scan (Index Only Scan)	table: customer; index: customer_pkey;	1	4.44	0.42
Hash Join		27	1384537.6	1079317.53
Aggregate		6666900	1143545.2	929996.06
Full Scan (Seq Scan)	table: payment;	6666900	109134	0
Transformation (Hash)		80	149320.46	149320.46
Unknown (Gather)		80	149320.46	2956.29
Nested Loops (Nested Loop)		33	148312.46	1956.29
Hash Join		33	148310.66	1956.13
Full Scan (Seq Scan)	table: reservation;	4166667	135416.67	0
Transformation (Hash)		1	1956.12	1956.12
Full Scan (Seq Scan)	table: pet;	1	1956.12	0
Unknown (Memoize)		1	0.18	0.16
Index Scan	table: reservationstatus; index: reservationstatus_pkey;	1	0.17	0.15

- Се забележува дека главниот проблем е Full Scan на табелата reservation и payment, и бидејќи и правиме join преку pet_id на reservation и join преку reservation_id на payment, можеме да ја индексираме по тие податоци.

```
CREATE INDEX idx_payment_reservation_id ON Payment (reservation_id);
CREATE INDEX idx_petID_reservation ON reservation(pet_id);
```

- Со помош на овој индекс времето за извршување значително се намалува на 2s 794ms.

```
[2026-07-01 17:13:54] postgres.public> select * from vw_my_pending_actions
                                where customer_id = 100286
[2026-07-01 17:13:56] 100 rows retrieved starting from 1 in 2 s 794 ms (execution: 2 s 455 ms, fetching: 339 ms)
```

Aggregate		6666900	332274.68	0.43
Index Scan	table: payment; index: idx_payment_reservation_id;	6666900	215603.93	0.43
Sort		80	4155.29	4155.09
Nested Loops (Nested Loop)		80	4152.56	8.13
Nested Loops (Nested Loop)		80	4149.57	7.97
Nested Loops (Nested Loop)		1	2603.95	0.42
Index Scan (Index Only)	table: customer; index: customer_pkey;	1	4.44	0.42
Full Scan (Seq Scan)	table: pet;	1	2599.5	0
Bitmap Index Scan (Bitmap)	table: reservation;	402	1541.6	7.55
Bitmap Index Scan	index: idx_petid_reservation;	402	7.45	0
Unknown (Memoize)		1	0.18	0.16
Index Scan	table: reservationstatus; index: reservationstatus_pkey;	1	0.17	0.15

View3: Room Occupancy

- Примарен филтер за погледот vw_room_occupancy ќе биде според check_in_date I check_out_date, а може да се користи и дополнитено со име на хотел.

- Примарен случај на употреба ќе е преглед на статусот на сите соби. За овој поглед ни се важни перформансите, бидејќи ќе биде користен секојдневно од клиенти. Иницијалното време за извршување на погледот е 3s 108ms.

```
[2026-05-20 10:32:46] postgres.public> SELECT * FROM vw_room_occupancy WHERE check_in_date>='2024-01-01' and check_out_date<='2024-01-31'  
[2026-05-20 10:32:49] 500 rows retrieved starting from 1 in 3 s 108 ms (execution: 2 s 770 ms, fetching: 338 ms)
```

- Ова не е прифатливо време за апликацијата, ама query-то може понатаму да се оптимизира со менување на логиката, со тоа што го изоставаме името на миленичето за кое е направена резервацијата бидејќи сметаме дека главната функционалност е да се прегледа статусот на собата.
- со наведените оптимизации времето на извршување е 441ms.

```
[2026-05-20 10:39:58] postgres.public> SELECT * FROM vw_room_occupancy WHERE check_in_date>='2024-01-01' and check_out_date<='2024-01-31'  
[2026-05-20 10:39:58] 500 rows retrieved starting from 1 in 441 ms (execution: 60 ms, fetching: 381 ms)
```

View4: Pet Profile

- Примарен филтер за погледот vw_pet_profile ќе биде според pet_id.
- Примарен случај на употреба ќе е преглед на профилот на милениче. За овој поглед ни се важни перформансите, бидејќи ќе биде користен секојдневно од клиенти. Иницијалното време за извршување на погледот е 369ms.

```
[2026-05-20 11:08:07] postgres.public> SELECT * FROM vw_pet_profile WHERE pet_id=80  
[2026-05-20 11:08:08] 1 row retrieved starting from 1 in 369 ms (execution: 12 ms, fetching: 357 ms)
```

- Ова е прифатливо време за апликацијата и поради тоа нема потреба од оптимизации.

View5: Reservation Financials

- Примарен филтер за погледот vw_reservation_financials ќе биде според reservation_date.
- Примарен случај на употреба ќе е преглед на профитот во одреден датум или период. За овој поглед не ни се важни перформансите,

бидејќи е аналитички поглед кој не е наменет за секојдневна употреба од корисници.

```
[2026-07-01 17:24:19] postgres.public> SELECT * FROM vw_reservation_financials -- analitika
                                WHERE reservation_date = CURRENT_DATE
[2026-07-01 17:30:51] 500 rows retrieved starting from 1 in 6 m 31 s 843 ms (execution: 6 m 31 s 483 ms, fetching: 360 ms)
```

- Погледот се извршува 6m 31s 843ms. Ова е прифатливо време за извршување поради тоа што е аналитички поглед, нема потреба од оптимизации.

View6: Today checkins and checkouts

- Примарен филтер за погледот vw_today_checkins_checkouts ќе биде според hotel_name.
- Примарен случај на употреба ќе е преглед на профитот на одреден датум. За овој поглед не ни се важни перформансите, бидејќи е аналитички.

```
[2026-07-01 17:38:00] postgres.public> SELECT * FROM vw_todays_checkins_checkouts
                                where hotel_name='Paw Palace'
[2026-07-01 17:38:01] 500 rows retrieved starting from 1 in 974 ms (execution: 579 ms, fetching: 395 ms)
```

- Погледот се извршува 974ms. Ова е прифатливо време за извршување, нема потреба од оптимизации.

View7: View available rooms

- Примарен филтер за погледот vw_available_rooms ќе биде според date.
- Примарен случај на употреба ќе е преглед на достапност на собите за одреден временски период.

```
[2026-07-01 18:21:38] postgres.public> SELECT * FROM vw_available_rooms
                                WHERE date > '2026-07-10' AND date < '2026-07-22'
                                ORDER BY room_number
[2026-07-01 18:21:38] 8 rows retrieved starting from 1 in 467 ms (execution: 16 ms, fetching: 451 ms)
```

- Погледот се извршува 467ms. Ова е прифатливо време за извршување, поради тоа нема потреба од оптимизации.

View8: View service schedule

- Примарен филтер за погледот vw_service_schedule ќе биде според date.

- Примарен случај на употреба ќе е преглед на сервисите кои треба да се извршени од вработените во одреден временски период.

```
[2026-07-01 18:23:35] postgres.public> SELECT * FROM vw_service_schedule WHERE scheduled_date = CURRENT_DATE
[2026-07-01 18:23:35] 500 rows retrieved starting from 1 in 512 ms (execution: 149 ms, fetching: 363 ms)
```

- Погледот се извршува 512ms. Ова е прифатливо време за извршување, поради тоа нема потреба од оптимизации.

View9: View medical history

- Примарен филтер за погледот vw_medical_history ќе биде според pet_id.
- Примарен случај на употреба ќе е преглед на медицинска историја на одредено милениче.

```
[2026-07-01 18:29:51] postgres.public> SELECT * FROM vw_medical_history
WHERE pet_id = 67890
ORDER BY record_date DESC
```

- Погледот се извршува 490ms. Ова е прифатливо време за извршување, поради тоа нема потреба од оптимизации.

View10: View employee workload

- Примарен филтер за погледот vw_employee_workload ќе биде според employee_id.
- Примарен случај на употреба ќе е преглед на колку работа има извршено даден вработен во одреден период.

```
[2026-07-01 19:13:46] postgres.public> SELECT * FROM vw_employee_workload
WHERE employee_id=7
[2026-07-01 19:13:47] 500 rows retrieved starting from 1 in 861 ms (execution: 527 ms, fetching: 334 ms)
```

- Погледот се извршува 861ms. Ова е прифатливо време за извршување, поради тоа нема потреба од оптимизации.

View11: View top customers

- Примарен филтер за погледот vw_top_customers ќе биде според hotel_name.
- Примарен случај на употреба ќе е преглед на кои корисници најмногу пари имаат потрошено. За овој поглед не ни се важни перформансите, бидејќи е аналитички.

```
[2026-07-01 19:30:38] postgres.public> SELECT * FROM vw_top_customers
      where hotel_name='Paw Palace'
[2026-07-01 19:30:43] 500 rows retrieved starting from 1 in 4 s 703 ms (execution: 4 s 373 ms, fetching: 330 m
```

- Погледот се извршува 4s 703ms. Ова е прифатливо време за извршување поради тоа што е аналитички поглед, нема потреба од оптимизации.

View13: guest stay summary

- Примарен филтер за погледот vw_guest_stay_summary ќе биде според pet_id.
- Примарен случај на употреба ќе е преглед на престој на милениче.

```
[2026-07-01 19:40:27] postgres.public> SELECT * FROM vw_guest_stay_summary
      WHERE pet_id = 8
      ORDER BY check_in_date DESC
[2026-07-01 19:40:28] 400 rows retrieved starting from 1 in 493 ms (execution: 32 ms, fetching: 461 ms)
```

- Погледот се извршува 493ms. Ова е прифатливо време за извршување, нема потреба од оптимизации.