Напредни бази на податоци

Фаза 3 - Индекси и оптимизација на прашалници

Проект:CinemaDB

Матеј Стошевски 233089

Петар Анѓелков 233280

Давид Стоиловски 233222

View1: vw_film_available_showtimes

1. Примарен филтер за погледот vw_film_available_showtimes ќе биде според име на филм (film), а исто така ќе се користи и според град и сала

```
advdb_202526l_prj_cdb_app.public> SELECT * FROM vw_film_available_showtimes WHERE film = '3 Idiots'
341 rows retrieved starting from 1 in 6 s 646 ms (execution: 6 s 251 ms, fetching: 395 ms)
```

2. Примарен случај на употреба ќе е пребарување на достапни термини за даден филм за да може да се направи резервација. За овој поглед ни се важни перформансите, бидејќи без него се губи време при извршување.
3. Иницијалното време за извршување на погледот е **3s 588ms** а пак за добивање на информации за даден филм беа потревни **6s 646ms**. Ова не е прифатливо време за апликацијата па затоа пристапуваме кон индексирање.

```
advdb_202526l_prj_cdb_app.public> SELECT * FROM vw_film_available_showtimes WHERE film = '3 Idiots'
341 rows retrieved starting from 1 in 6 s 646 ms (execution: 6 s 251 ms, fetching: 395 ms)
```

```
[2026-05-09 20:16:43] advdb_202526l_prj_cdb_app> --VIEW 1 za daden film vo koja sala i koga ima slobodni mesta
                                                 CREATE VIEW vw_film_available_showtimes AS
                                                 SELECT
                                                     m.movie_id,
                                                     m.title                         AS film,
                                                     m.duration                      AS trajanje_min,
                                                     c.name                          AS kino,
                                                     c.city                          AS grad,
                                                     h.hall_id,
                                                     h.name                          AS sala,
                                                     s.showtime_id,
                                                     s.start_time,
                                                     s.end_time,
                                                     s.base_price,
                                                     h.capacity                      AS vkupni_mesta,
                                                     COUNT(t.ticket_id)              AS zafateni_mesta,
                                                     h.capacity - COUNT(t.ticket_id) AS slobodni_mesta
                                                 FROM Movie m
                                                 JOIN Showtime s   ON s.movie_id  = m.movie_id
                                                 JOIN Hall h       ON h.hall_id   = s.hall_id
                                                 JOIN Cinema c     ON c.cinema_id = h.cinema_id
                                                 LEFT JOIN Ticket t ON t.showtime_id = s.showtime_id
                                                 GROUP BY
                                                     m.movie_id, m.title, m.duration,
                                                     c.name, c.city,
                                                     h.hall_id, h.name, h.capacity,
                                                     s.showtime_id, s.start_time, s.end_time, s.base_price
                                                 HAVING h.capacity - COUNT(t.ticket_id) > 0
                                                 ORDER BY m.title, s.start_time
[2026-05-09 20:16:47] completed in 3 s 588 ms
```

4. Најбавните операции се Full Scan на табела Ticket (3,776,753 редови, cost 133,990) и Full Scan на табела Showtime (568,345 редови, cost 20,366) и тие можат да се подобрат со индекси. Времето изминато во извршување на операциите insert и update изнесува **1,213ms** и **2,181ms**.

```
CREATE INDEX idx_ticket_showtime ON Ticket(showtime_id DESC NULLS LAST);
CREATE INDEX idx_showtime_movie ON Showtime(movie_id DESC NULLS LAST);
```

Output | Plan

Operation	Params	Rows	Total Cost	Startup Cost	Raw Desc
Select					JIT = {"Functions":44,"Options":{"Inlin...
Sort		1029	171777.41	171774.84	Parallel Aware = false;Async Capable...
Aggregate		1029	171723.35	171258.55	Strategy = Sorted;Partial Mode = Sim...
Unknown (Gather Merge)		3087	171628.17	171258.55	Parent Relationship = Outer;Parallel ...
Sort		772	170260.42	170258.49	Parent Relationship = Outer;Parallel ...
Merge Join		772	170221.46	170206.02	Parent Relationship = Outer;Parallel ...
Sort		772	170185.06	170183.13	Parent Relationship = Outer;Parallel ...
Hash Join		772	170146.11	21990.27	Parent Relationship = Outer;Parallel ...
Full Scan (Seq Scan)	table: ticket;	3776753	133990.53	0.0	Parent Relationship = Outer;Parallel ...
Transformation (Hash)		116	21988.82	21988.82	Parent Relationship = Inner;Parallel A...
Hash Join		116	21988.82	129.17	Parent Relationship = Outer;Parallel ...
Full Scan (Seq Scan)	table: showtime;	568345	20366.45	0.0	Parent Relationship = Outer;Parallel ...
Transformation (Hash)		1	129.16	129.16	Parent Relationship = Inner;Parallel A...
Full Scan (Seq Scan)	table: movie;	1	129.16	0.0	Parent Relationship = Outer;Parallel ...
Sort		200	23.39	22.89	Parent Relationship = Inner;Parallel A...
Hash Join		200	15.24	10.68	Parent Relationship = Outer;Parallel ...
Full Scan (Seq Scan)	table: hall;	200	4.0	0.0	Parent Relationship = Outer;Parallel ...
Transformation (Hash)		30	10.3	10.3	Parent Relationship = Inner;Parallel A...
Full Scan (Seq Scan)	table: cinema;	30	10.3	0.0	Parent Relationship = Outer;Parallel ...

```
[2026-05-09 20:30:22] advdb_202526l_prj_cdb_app.public> INSERT INTO Ticket (showtime_id, seat_id, reservation_id, price, purchase_date)
                                                        VALUES (1, 1, 1, 500, CURRENT_DATE)
[2026-05-09 20:30:23] 1 row affected in 1 s 213 ms
[2026-05-09 20:30:49] advdb_202526l_prj_cdb_app.public> UPDATE Ticket SET price = 600 WHERE showtime_id = 1
[2026-05-09 20:30:51] 7 rows affected in 2 s 181 ms
```

5. Времето изминато во извршување на query-то со индекси изнесува **152ms** и времето потребно за добивање на информациите за конкретрен филм е **203ms**, и тоа е прифатливо време.

```
[2026-05-09 20:26:20] advdb_202526l_prj_cdb_app.public> --VIEW 1 za daden film vo koja sala i koga ima slobodni mesta
                                                        CREATE VIEW vw_film_available_showtimes AS
                                                        SELECT
                                                            m.movie_id,
                                                            m.title                             AS film,
                                                            m.duration                          AS trajanje_min,
                                                            c.name                              AS kino,
                                                            c.city                              AS grad,
                                                            h.hall_id,
                                                            h.name                              AS sala,
                                                            s.showtime_id,
                                                            s.start_time,
                                                            s.end_time,
                                                            s.base_price,
                                                            h.capacity                          AS vkupni_mesta,
                                                            COUNT(t.ticket_id)                  AS zafateni_mesta,
                                                            h.capacity - COUNT(t.ticket_id)     AS slobodni_mesta
                                                        FROM Movie m
                                                        JOIN Showtime s   ON s.movie_id  = m.movie_id
                                                        JOIN Hall h       ON h.hall_id   = s.hall_id
                                                        JOIN Cinema c     ON c.cinema_id = h.cinema_id
                                                        LEFT JOIN Ticket t ON t.showtime_id = s.showtime_id
                                                        GROUP BY
                                                            m.movie_id, m.title, m.duration,
                                                            c.name, c.city,
                                                            h.hall_id, h.name, h.capacity,
                                                            s.showtime_id, s.start_time, s.end_time, s.base_price
                                                        HAVING h.capacity - COUNT(t.ticket_id) > 0
                                                        ORDER BY m.title, s.start_time
[2026-05-09 20:26:20] completed in 152 ms
```

```
advdb_202526l_prj_cdb_app.public> SELECT * FROM vw_film_available_showtimes WHERE film = '3 Idiots'
341 rows retrieved starting from 1 in 1 s 203 ms (execution: 282 ms, fetching: 921 ms)
```

6. Времето изминато во извршување на операциите insert и update по индексирање изнесува **539ms** и **1,514ms**.

```
[2026-05-09 20:32:06] advdb_202526l_prj_cdb_app.public> INSERT INTO Ticket (showtime_id, seat_id, reservation_id, price, purchase_date)
                                                        VALUES (1, 1, 1, 500, CURRENT_DATE)
[2026-05-09 20:32:07] 1 row affected in 539 ms
[2026-05-09 20:32:15] advdb_202526l_prj_cdb_app.public> UPDATE Ticket SET price = 600 WHERE showtime_id = 1
[2026-05-09 20:32:16] 8 rows affected in 1 s 514 ms
```

7. Иако погледот користи GROUP BY и COUNT, тој е од **функционален тип** бидејќи примарната цел е корисникот да пронајде конкретен филм и да види во која сала и кога има слободни места за да направи резервација. Агрегацијата се користи само за филтрирање на проекции со слободни места преку HAVING клаузулата, а не за аналитички цели. Се обидовме да го преуредиме query-то без GROUP BY и COUNT користејќи subquery и NOT EXISTS, меѓутоа тие верзии се покажаа значително побавни бидејќи за секоја проекција посебно се извршува операција врз Ticket табелата. Затоа оригиналната верзија со GROUP BY и COUNT е најоптималното решение.

View2: vw_user_reservations

1. Примарен филтер за погледот vw_user_reservations ќе биде според user_id, а исто така ќе може да се пребарува и според email.
2. Примарен случај на употреба ќе е преглед на сите резервации на даден корисник — кој филм, во кое кино, која сала, кое седиште, и дали е платено.

   За овој поглед ни се важни перформансите, бидејќи без него се губи време при извршување.
3. Иницијалното време за извршување на погледот е **5s 473ms**. Ова не е прифатливо време за апликацијата па затоа пристапуваме кон

индексирање.

```
[2026-05-22 12:13:04] cinemaDB.public> EXPLAIN ANALYZE
                                       SELECT * FROM vw_user_reservations WHERE user_id = 42
[2026-05-22 12:13:09] 90 rows retrieved starting from 1 in 5 s 473 ms (execution: 5 s 120 ms, fetching: 353 ms)
```

4. Најбавните операции се Full Scan на табела Reservation, Full Scan на табела Ticket и Full Scan на табела ReservationPayment.

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Unknown (Gather Merge)		153	492908.13	492890.31
Sort		64	491890.44	491890.28
Hash Join		64	491888.36	355894.07
Full Scan (Seq Scan)	table: reservationpayment;	4168415	120362.15	0
Transformation (Hash)		64	355893.27	355893.27
Nested Loops (Nested Loop)		64	355893.27	179513.63
Nested Loops (Nested Loop)		64	355890.7	179513.48
Hash Join		64	355871.02	179513.19
Hash Join		64	355860.16	179502.51
Nested Loops (Nested Loop)		64	355853.49	179496.01
Nested Loops (Nested Loop)		64	355834.28	179495.73
Nested Loops (Nested Loo		64	355296.34	179495.3
Hash Join		64	355287.22	179495.01
Full Scan (Seq Scan)	table: ticket;	6298155	159259.55	0
Transformation (Hash)		42	179494.48	179494.48
Full Scan (Seq Sc	table: reservation;	42	179494.48	0
Temporary (Materialize)		1	8.32	0.29
Index Scan	table: cinemauser; index: cinemauser_pkey;	1	8.31	0.29
Index Scan	table: showtime; index: showtime_pkey;	1	8.41	0.43
Index Scan	table: movie; index: movie_pkey;	1	0.3	0.28
Transformation (Hash)		200	4	4
Full Scan (Seq Scan)	table: hall;	200	4	0
Transformation (Hash)		30	10.3	10.3
Full Scan (Seq Scan)	table: cinema;	30	10.3	0
Index Scan	table: seat; index: seat_pkey;	1	0.31	0.29

и тие можат да се подобрат со индекси.Времето изминато во извршување на операциите insert и update изнесува:

```
[2026-05-22 12:44:39] cinemaDB.public> EXPLAIN ANALYZE
                                       INSERT INTO Reservation (user_id, showtime_id, employee_id)
                                       VALUES (42, 1, 1)
[2026-05-22 12:44:39] 9 rows retrieved starting from 1 in 348 ms (execution: 10 ms, fetching: 338 ms)
[2026-05-22 12:44:55] cinemaDB.public> EXPLAIN ANALYZE
                                       UPDATE Reservation SET status = 'CONFIRMED' WHERE user_id = 42
[2026-05-22 12:44:56] 8 rows retrieved starting from 1 in 1 s 454 ms (execution: 1 s 113 ms, fetching: 341 ms)
```

5. Времето изминато во извршување на query-то со индекси изнесува **410ms**, и тоа е прифатливо време.

```
CREATE INDEX idx_reservation_user ON Reservation(user_id);
CREATE INDEX idx_ticket_reservation ON Ticket(reservation_id);
CREATE INDEX idx_payment_reservation ON ReservationPayment(reservation_id);
```

```
[2026-05-22 13:01:33] cinemaDB.public> EXPLAIN ANALYZE
                                       SELECT * FROM vw_user_reservations WHERE user_id = 42
[2026-05-22 13:01:33] 85 rows retrieved starting from 1 in 410 ms (execution: 26 ms, fetching: 384 ms)
```

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Sort		153	3510.9	3510.51
Nested Loops (Nested Loop)		153	3504.96	7.84
Nested Loops (Nested Loop)		153	2643.8	7.39
Nested Loops (Nested Loop)		153	2639.02	7.24
Nested Loops (Nested Loop)		153	2591.96	6.95
Nested Loops (Nested Loop)		153	2568.4	6.8
Nested Loops (Nested Loop)		101	1306.93	6.37
Nested Loops (Nested Loop)		101	1290.53	6.22
Nested Loops (Nested Lo		101	1260.21	5.94
Nested Loops (Nested		101	411.27	5.51
Index Scan	table: cinemauser; index: cinemauser_pkey;	1	8.31	0.29
Bitmap Index Scan	table: reservation;	101	401.95	5.22
Bitmap Index Sc	index: idx_reservation_user;	101	5.19	0
Index Scan	table: showtime; index: showtime_pkey;	1	8.41	0.43
Index Scan	table: movie; index: movie_pkey;	1	0.3	0.28
Index Scan	table: hall; index: hall_pkey;	1	0.16	0.14
Index Scan	table: ticket; index: idx_ticket_reservation;	2	12.47	0.43
Unknown (Memoize)		1	0.41	0.15
Index Scan	table: cinema; index: cinema_pkey;	1	0.4	0.14
Index Scan	table: seat; index: seat_pkey;	1	0.31	0.29
Unknown (Memoize)		1	0.17	0.15
Index Scan	table: seat_type; index: seat_type_pkey;	1	0.16	0.14
Unknown (Memoize)		1	8.46	0.45
Index Scan	table: reservationpayment; index: idx_payment_reservation;	1	8.45	0.43

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

```
[2026-05-22 13:06:05] cinemaDB.public> EXPLAIN ANALYZE
                                       INSERT INTO Reservation (user_id, showtime_id, employee_id)
                                       VALUES (42, 1, 1)
[2026-05-22 13:06:05] 9 rows retrieved starting from 1 in 361 ms (execution: 10 ms, fetching: 351 ms)
[2026-05-22 13:06:08] cinemaDB.public> EXPLAIN ANALYZE
                                       UPDATE Reservation SET status = 'CONFIRMED' WHERE user_id = 42
[2026-05-22 13:06:08] 12 rows retrieved starting from 1 in 368 ms (execution: 9 ms, fetching: 359 ms)
```

VIEW 3: vw_order_details

1. Примарен филтер за погледот vw_order_details ќе биде според ID на нарачка (order_id), а исто така може да се користи и според корисник за пребарување на историјата на нарачки. Погледот се користи за прикажување на детали за одредена нарачка - сите производи, количина, цена, информација за корисникот...

2. Примарен случај на употреба ќе е пребарување на детали за конкретна нарачка со цел прикажување или потврда на нарачка или историја на корисник. За овој поглед ни се важни перформансите, бидејќи

корисниците очекуваат брз одговор при проверка на своите нарачки, а касиерите при печатење на фискални сметки.

3. Иницијалното време за извршување на погледот е 947 ms (6 редови), што е прифатливо за мала количина на подаци. Времето потребно за пребарување на детали за конкретна нарачка (order_id = 432) беше 947 ms. Времето изминато во извршување на операциите insert и update пред индексирање изнесува **86 ms** и **451 ms** соодветно, каде што UPDATE е особено бавен.

```
[2026-05-14 00:43:11] advdb_202526l_prj_cdb_app.public> SELECT * FROM vw_order_details WHERE order_id = 432
[2026-05-14 00:43:12] 6 rows retrieved starting from 1 in 947 ms (execution: 555 ms, fetching: 392 ms)
[2026-05-14 00:43:43] advdb_202526l_prj_cdb_app.public> INSERT INTO Order_Product (order_id, product_id, quantity, price_at_order)
                                                        VALUES (432,5,2,150)
[2026-05-14 00:43:43] 1 row affected in 86 ms
[2026-05-14 00:43:49] advdb_202526l_prj_cdb_app.public> UPDATE Order_Product SET quantity = 3, price_at_order = 160 WHERE order_id = 432 AND product_id = 5
[2026-05-14 00:43:49] 1 row affected in 451 ms
```

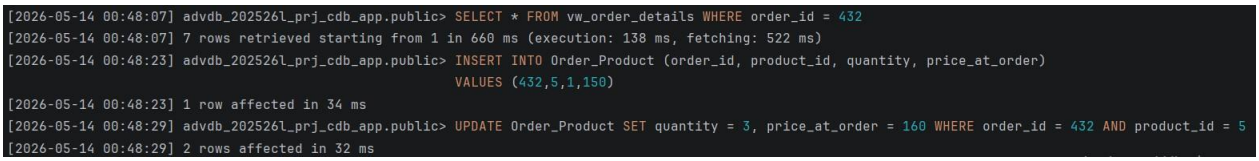
4. Најбавната операција е **Full Scan** на табела Order_Product (1 ред, cost 42,725.5) која содржи поврзување на производи со нарачки. Овој Full Scan беше главниот проблем затоа што без индекс на order_id, системот морал да скенира сите редови пред да ги филтрира по конкретна нарачка. Исто така, Nested Loops на CinemaUser и Employee табели влијаеше на вкупното време. Времето за JOIN операциите достигна и до **43,755 ms** cost.

Output | Za TEST na VIEW 3 | Query Plan | Za TEST na VIEW 4

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Unknown (Gather)		4	43755.17	1001.16
Nested Loops (Nested Loc		1	42754.77	1.16
Nested Loops (Nested		1	42750.58	1
Nested Loops (Nest		1	42742.27	0.72
Full Scan (Seq Sc	table: order_product;	1	42725.5	0
Nested Loops (N		1	16.76	0.72
Index Scan	table: cinemaorder; index: cinemaorder_pkey;	1	8.45	0.43
Index Scan	table: cinemauser; index: cinemauser_pkey;	1	8.31	0.29
Unknown (Memoize)		1	8.3	0.28
Index Scan	table: employee; index: employee_pkey;	1	8.29	0.27
Unknown (Memoize)		1	4.17	0.15
Index Scan	table: product; index: product_pkey;	1	4.16	0.14

5. По имплементирање на еден единствен индекс idx_order_product_order_id на табелата Order_Product, времето за SELECT операција се намали на **660 ms (30% подобрување)**. Овој еден индекс беше доволен да се подобри ефикасноста на целиот поглед. Времето за INSERT и UPDATE операции се намалиа значително по индексирање.

6. Времето изминато во извршување на операциите insert и update по индексирање изнесува **34 ms** и **32 ms** соодветно. INSERT операцијата се подобри за **60% (од 86 ms на 34 ms)**, а UPDATE операцијата за **93% (од 451 ms на 32 ms)**. Овие подобрувања покажуваат колку критично е индексирањето за операциите што работат со Order_Product табелата.



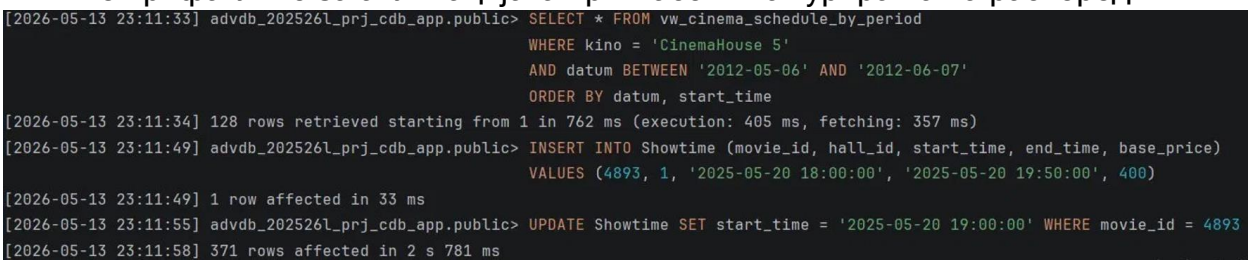
7. Заклучок: Додавањето на еден единствен индекс idx_order_product_order_id беше доволно за оптимизација на целиот поглед. SELECT перформансите се подобрија за 30% (947 → 660 ms), INSERT за 60% (86 → 34 ms), а UPDATE за 93% (451 → 32 ms). Ова докажува колку моќен може да биде правилниот индекс - еден индекс на правото место направи голема разлика. Системот сега е способен да работи со нарачките многу поефикасно и брзо, што е критично за корисничкото искуство на касата и раководењето на нарачките.

VIEW 4: vw_cinema_schedule_by_period

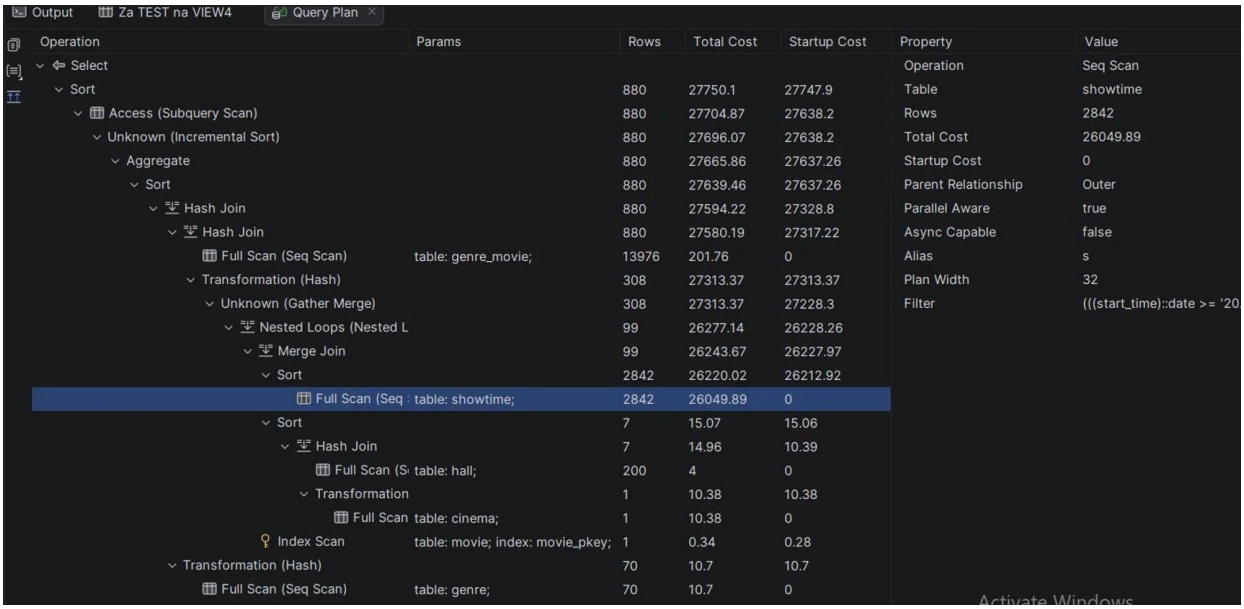
1. Примарен филтер за погледот vw_cinema_schedule_by_period ќе биде според кино, а исто така ќе се користи и според датум. Погледот се користи за прикажување на целиот распоред на проекции по кино во одреден ден или недела.

2. Примарен случај на употреба ќе е пребарување на достапни филмови во конкретно кино за одреден временски период, со цел управување на распоредот и давање на информации до корисниците. За овој поглед ни се важни перформансите, бидејќи е потребно брзо прикажување на информации за да се подобри корисничкото искуство.

3. Иницијалното време за извршување на погледот е **762 ms**, а времето потребно за пребарување на распоред за конкретно кино (CinemaHouse 5) во одреден период беше **706 ms**. Времето изминато во извршување на операцијата update (промена на време на проекција) беше **2,700 ms**, што не е прифатливо за апликацијата при масовни ажурирања на распоред.



4. Најбавната операција е Full Scan на табела Showtime (2,842 редови, cost 26,049) која се користи за пристап до сите проекции. Исто така, Full Scan на табела Genre_Movie (13,976 редови, cost 201) влијае на времето за извршување поради необходноста од агрегирање на жанрови. UPDATE операцијата беше скапа затоа што без индекс морала да скенира сите редови пред да ги ажурира. Времето изминато во извршување на операциите insert и update пред индексирање изнесува **33 ms и 2,700 ms** соодветно.



5. По искористување на индексот idx_showtime_movie кој беше направен во View 1 на табелата Showtime, времето за извршување на погледот се намали на **706 ms** (мало подобрување), што е прифатливо. Времето потребно за пребарување на распоред за конкретно кино остана слично (706 ms), бидејќи главниот филтер е по кино а не по movie_id.

```
[2026-05-13 23:16:17] advdb_202526l_prj_cdb_app.public> SELECT * FROM vw_cinema_schedule_by_period
                                                        WHERE kino = 'CinemaHouse 5'
                                                        AND datum BETWEEN '2012-05-06' AND '2012-06-07'
                                                        ORDER BY datum, start_time
[2026-05-13 23:16:18] 128 rows retrieved starting from 1 in 706 ms (execution: 360 ms, fetching: 346 ms)
[2026-05-13 23:16:34] advdb_202526l_prj_cdb_app.public> UPDATE Showtime SET start_time = '2025-05-20 19:00:00' WHERE movie_id = 4893
[2026-05-13 23:16:34] 371 rows affected in 52 ms
```

6. Времето изминато во извршување на операциите insert и update по индексирање е 33 ms и 52 ms соодветно. UPDATE операцијата се подобри за 98% (од 2,700 ms на 52 ms), што е драматично подобрување. Индексот idx_showtime_movie_id значајно ја зголеми ефикасноста на UPDATE операциите кои филтрираат по movie_id.

7. Заклучок: Искористувањето на индексот idx_showtime_movie беше критично за оптимизација на UPDATE операциите на Showtime табелата. Иако SELECT перформансите минимално се подобрија, UPDATE

перформансите се зголемија за 98%, што го прави системот многу подобар при ажурирање

на податоци. Индексот по Genre_Movie(movie_id) не покажа видливо подобрување, затоа што Full Scan на 13,976 редови има релативно мала цена во споредба со другите операции, па затоа индексот не беше имплементиран. Избраниот индекс систем е оптимален за перформансите на овој поглед.

View5: vw_movie_reviews_summary

1. Примарен филтер за погледот vw_movie_reviews_summary е **movie_id**, а исто така може да се пребарува и според **title**.
2. Примарен случај на употреба е преглед на резимето на рецензии за даден филм – просечна оцена, вкупен број на рецензии, максимална и минимална оцена и жанрови.
3. Овој поглед е имплементиран како **Materialized View** поради тоа што нема да се менува често, на корисникот нема да му смени многу доколку рецензијата се смени со rating од 4,7 на 4,8. Kaj Materialized View резултатите се чуваат физички на диск, што го елиминира потребата од агрегација на милиони редови при секое барање. Креирањето на погледот потрае **3 мин 23 сек**, што е нормално за ваква количина на податоци.

```
[2026-05-15 22:13:54] cinemaDB.public> CREATE MATERIALIZED VIEW vw_movie_reviews_summary AS
                                       SELECT
                                           m.movie_id,
                                           m.title,
                                           m.release_year,
                                           m.language,
                                           ROUND(AVG(r.rating), 2)     AS avg_rating,
                                           COUNT(r.review_id)          AS total_reviews,
                                           MAX(r.rating)               AS max_rating,
                                           MIN(r.rating)               AS min_rating,
                                           STRING_AGG(DISTINCT g.name, ', ') AS genres
                                       FROM Movie m
                                       LEFT JOIN Review r      ON r.movie_id = m.movie_id
                                       LEFT JOIN Genre_Movie gm ON gm.movie_id = m.movie_id
                                       LEFT JOIN Genre g        ON g.genre_id = gm.genre_id
                                       GROUP BY m.movie_id, m.title, m.release_year, m.language
[2026-05-15 22:17:18] 4,893 rows affected in 3 m 23 s 815 ms
```

4. Времето за извршување на SELECT прашалникот врз Materialized View е **385ms** по movie_id и **358ms** по title, што е прифатливо време.
5. Нема потреба од индексирање – Materialized View веќе ги чува пресметаните агрегати физички, па при SELECT се чита директно од зачуваниот резултат без скенирање на оригиналните табели.
6. Бидејќи рецензиите не се менуваат често, погледот не се освежува

автоматски – наместо тоа, се освежува рачно по потреба со командата **REFRESH MATERIALIZED VIEW vw_movie_reviews_summary**, на пример откако ќе се додадат поголем број нови рецензии во системот или на крајот на денот, со што се гарантира дека корисниците гледаат ажурирани просечни оцени без да се жртвуваат перформансите при секое барање.

```
[2026-05-15 22:18:08] cinemaDB.public> SELECT * FROM vw_movie_reviews_summary WHERE movie_id = 1
[2026-05-15 22:18:08] 1 row retrieved starting from 1 in 385 ms (execution: 5 ms, fetching: 380 ms)
[2026-05-15 22:18:41] cinemaDB.public> SELECT * FROM vw_movie_reviews_summary WHERE title = 'Inception'
[2026-05-15 22:18:42] 1 row retrieved starting from 1 in 358 ms (execution: 4 ms, fetching: 354 ms)
[2026-05-15 22:19:30] cinemaDB.public> REFRESH MATERIALIZED VIEW vw_movie_reviews_summary
[2026-05-15 22:22:48] completed in 3 m 18 s 489 ms
```

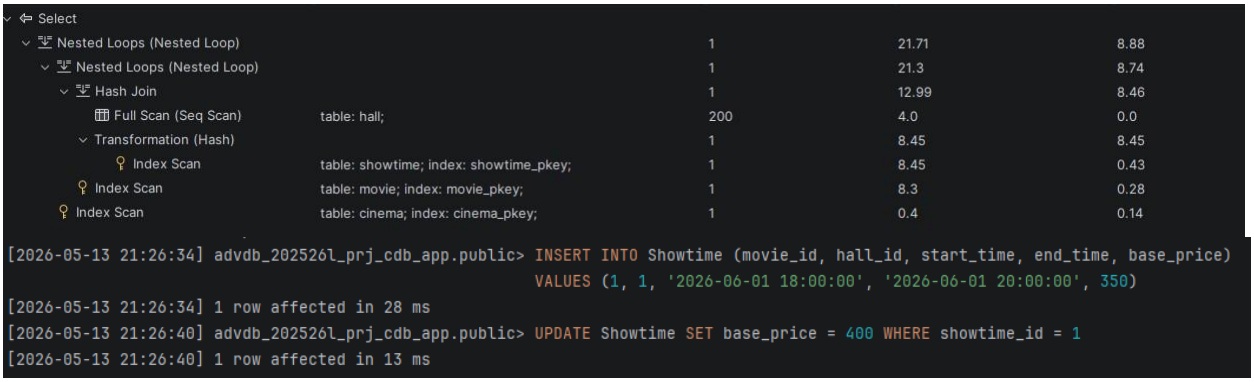
View6: vw_showtime_schedule

1. Примарен филтер за погледот vw_showtime_schedule ќе биде според showtime_id, а исто така ќе може да се пребарува и според име на филм и име на кино.
2. Примарен случај на употреба ќе е преглед на детали за конкретна проекција - кое кино, која сала, кога започнува и завршува, и која е основната цена на билет. За овој поглед ни се важни перформансите, бидејќи без него се губи време при извршување.

```
[2026-05-13 21:23:19] advdb_202526l_prj_cdb_app.public> CREATE VIEW vw_showtime_schedule AS
                                                        SELECT
                                                            s.showtime_id,
                                                            m.title AS movie_title,
                                                            m.duration,
                                                            c.name AS cinema_name,
                                                            h.name AS hall_name,
                                                            s.start_time,
                                                            s.end_time,
                                                            s.base_price,
                                                            h.capacity AS total_seats
                                                        FROM Showtime s
                                                        JOIN Movie m ON m.movie_id = s.movie_id
                                                        JOIN Hall h ON h.hall_id = s.hall_id
                                                        JOIN Cinema c ON c.cinema_id = h.cinema_id
[2026-05-13 21:23:19] completed in 16 ms
```

```
[2026-05-13 21:24:17] advdb_202526l_prj_cdb_app.public> SELECT * FROM vw_showtime_schedule WHERE showtime_id = 3
[2026-05-13 21:24:18] 1 row retrieved starting from 1 in 335 ms (execution: 17 ms, fetching: 318 ms)
```

3. Иницијалното време за извршување на погледот е **335ms**. Ова е прифатливо време за апликацијата.
4. Нема потреба од правење план на извршување, бидејќи времето е задоволително. Сите joins се извршуваат преку автоматски Primary Key индекси (showtime_pkey, movie_pkey, cinema_pkey). Единствениот Full Scan е на hall табелата која има само 200 редови и не претставува проблем. Времето на операции insert и update изнесува **28ms** и **13ms**.



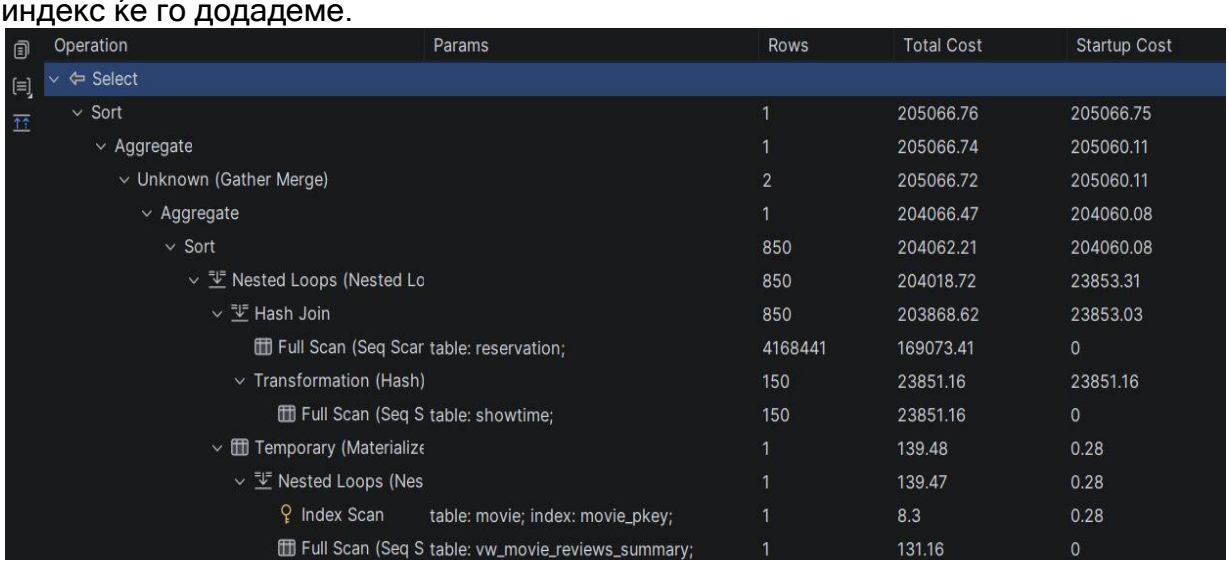
5. Нема потреба да се преуреди прашалникот.
6. Време на извршување на операциите останува исто – **28ms** и **13ms**.

View7: vw_popular_movies

1. Примарен филтер за погледот vw_popular_movies ќе биде според movie_id, а исто така ќе може да се пребарува и според наслов на филм.
2. Примарен случај на употреба ќе е преглед на популарноста на даден филм - вкупен број на резервации, просечна оцена и број на рецензии. За овој поглед ни се важни перформансите, бидејќи без него се губи време при извршување.
3. Иницијалното време за извршување на погледот е **1s 376ms**. Ова не е прифатливо време за апликацијата па затоа пристапуваме кон индексирање.

```
[2026-05-22 16:43:33] cinemaDB.public> EXPLAIN ANALYZE
                                       SELECT * FROM vw_popular_movies WHERE movie_id = 1
[2026-05-22 16:43:35] 51 rows retrieved starting from 1 in 1 s 729 ms (execution: 1 s 376 ms, fetching: 353 ms)
```

4. Најбавните операции се Full Scan на табела Reservation (4,168,441 редови, cost 169,073) и Full Scan на табела Showtime (150 редови, cost 23,851) и тие можат да се подобрат со индекси. Може да се искористи веќе постоечкиот индекс од погледот 2 (за user_id во табелата reservation), додека вториот индекс ќе го додадеме.



Времето изминато во извршување на операциите INSERT и UPDATE пред индексирање изнесува **11ms** и **1s 797ms**.

```
[2026-05-22 16:46:55] cinemaDB.public> EXPLAIN ANALYZE
                                       INSERT INTO Reservation (user_id, showtime_id, employee_id, reservation_date, status)
                                       VALUES (1, 1, 1, CURRENT_DATE, 'PENDING')
[2026-05-22 16:46:56] 9 rows retrieved starting from 1 in 574 ms (execution: 11 ms, fetching: 563 ms)
[2026-05-22 16:47:18] cinemaDB.public> EXPLAIN ANALYZE
                                       UPDATE Reservation SET status = 'CONFIRMED'
                                       WHERE user_id = 1 AND showtime_id = 1
[2026-05-22 16:47:20] 8 rows retrieved starting from 1 in 2 s 136 ms (execution: 1 s 797 ms, fetching: 339 ms)
```

5. Погледот црпи податоци од vw_movie_reviews_summary, кој е имплементиран како Materialized View и ги чува резултатите физички на диск. Благодарение на тоа, агрегацијата на рецензиите не се извршува при секое барање туку се чита директно од меморија.

6. По примена на постоечките индекси idx_reservation_showtime на табелата Reservation(showtime_id) и idx_reservation_user на табелата Reservation(user_id), времето за SELECT се намали на 503ms, а cost на погледот се намали од 205,066 на 29,293.

Output | Query Plan | Result 74

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Sort		1	29293.77	29293.76
Aggregate		1	29293.75	29287.12
Unknown (Gather Merge)		2	29293.73	29287.12
Aggregate		1	28293.48	28287.09
Sort		850	28289.22	28287.09
Nested Loops (Nested Loop)		850	28245.74	0.72
Nested Loops (Nested Loop)		150	23992.51	0.28
Full Scan (Seq Scan)	table: showtime;	150	23851.16	0
Temporary (Materialize)		1	139.48	0.28
Nested Loops (Nested Loop)		1	139.47	0.28
Index Scan	table: movie; index: movie_pkey;	1	8.3	0.28
Full Scan (Seq Scan)	table: vw_movie_reviews_summary;	1	131.16	0
Index Scan	table: reservation; index: idx_reservation_sh...	6	28.29	0.43

7. Времето изминато во извршување на операциите INSERT и UPDATE по индексирање изнесува 12ms и 9ms.

```
[2026-05-22 16:52:45] cinemaDB.public> EXPLAIN ANALYZE
                                       INSERT INTO Reservation (user_id, showtime_id, employee_id, reservation_date, status)
                                       VALUES (1, 1, 1, CURRENT_DATE, 'PENDING')
[2026-05-22 16:52:45] 9 rows retrieved starting from 1 in 540 ms (execution: 12 ms, fetching: 528 ms)
[2026-05-22 16:52:51] cinemaDB.public> EXPLAIN ANALYZE
                                       UPDATE Reservation SET status = 'CONFIRMED'
                                       WHERE user_id = 1 AND showtime_id = 1
[2026-05-22 16:52:51] 18 rows retrieved starting from 1 in 395 ms (execution: 9 ms, fetching: 386 ms)
```

View8: vw_presentation_rights

1. Примарен филтер за погледот vw_presentation_rights ќе биде според име на филм, а исто така ќе може да се пребарува и според кино.
2. Примарен случај на употреба ќе е преглед на кои кина имаат право да прикажуваат конкретен филм. За овој поглед ни се важни перформансите, бидејќи без него се губи време при извршување.

```
[2026-05-13 22:25:30] advdb_202526l_prj_cdb_app.public> SELECT * FROM vw_presentation_rights WHERE film = 'The Godfather'
[2026-05-13 22:25:31] 55 rows retrieved starting from 1 in 378 ms (execution: 47 ms, fetching: 331 ms)
```

3. Иницијалното време за извршување на погледот е **378ms**. Ова е прифатливо време за апликацијата па затоа нема да имаме потреба од индексирање.
4. Нема потреба од правење план на извршување, бидејќи времето е задоволително. Иако постои Full Scan на PresentationRights (152,636 редови, cost 3,179) и Movie (cost 129), Cinema табелата користи автоматски Primary Key индекс (cinema_pkey). Времето е прифатливо поради едноставноста на прашалникот - само два прости JOIN-ови без агрегација. За вакви едноставни прашалници PostgreSQL одлучува дека Full Scan е поефикасно од вештачки индекс. Времето на операции insert и update изнесува **11ms** и **36ms**.

Select				
Unknown (Gather)		53	4719.64	1129.31
Nested Loops (Nested Loop)		31	3714.34	129.31
Hash Join		31	3709.55	129.17
Full Scan (Seq Scan)	table: presentationrights;	152636	3179.36	0.0
Transformation (Hash)		1	129.16	129.16
Full Scan (Seq Scan)	table: movie;	1	129.16	0.0
Index Scan	table: cinema; index: cinema_pkey;	1	0.16	0.14

```
[2026-05-13 22:25:42] advdb_202526l_prj_cdb_app.public> INSERT INTO PresentationRights (movie_id, cinema_id, start_date, end_date)
                                                        VALUES (1, 1, '2026-01-01', '2026-12-31')
[2026-05-13 22:25:42] 1 row affected in 11 ms
[2026-05-13 22:25:46] advdb_202526l_prj_cdb_app.public> UPDATE PresentationRights SET end_date = '2027-12-31' WHERE movie_id = 1 AND cinema_id = 1
[2026-05-13 22:25:46] 2 rows affected in 36 ms
```

5. Време на извршување на операциите останува исто – **11ms** и **36ms**.

View9: vw_monthly_revenue

6. Примарен филтер за погледот vw_monthly_revenue ќе биде според cinema_name, а исто така ќе може да се пребарува и според месец.

```
[2026-05-09 23:18:50] advdb_202526l_prj_cdb_app.public> SELECT * FROM vw_monthly_revenue WHERE cinema_name = 'CinemaHouse 1'
[2026-05-09 23:19:05] 297 rows retrieved starting from 1 in 14 s 664 ms (execution: 14 s 242 ms, fetching: 422 ms)
```

7. Овој поглед е од **аналитичен тип** — врши агрегација на податоци (SUM, COUNT) групирани по месец и кино. Се користи за преглед на месечните приходи по кино. Кај аналитични прашалници перформансите не се критични бидејќи се извршуваат поретко и се очекува поголемо време на извршување.
8. Иницијалното време за извршување на погледот е **14s 664ms**. Иако ова е долго време, тоа е очекувано за аналитички прашалник кој агрегира милиони редови од ReservationPayment табелата. Нема потреба од индексирање.

```
[2026-05-09 23:18:50] advdb_202526l_prj_cdb_app.public> SELECT * FROM vw_monthly_revenue WHERE cinema_name = 'CinemaHouse 1'
[2026-05-09 23:19:05] 297 rows retrieved starting from 1 in 14 s 664 ms (execution: 14 s 242 ms, fetching: 422 ms)
```

9. Најбавните операции се Full Scan на ReservationPayment и Reservation табелите, меѓутоа бидејќи се работи за аналитичко query кое мора да ги процесира сите записи за да ги пресмета агрегатите, индексите не би помогнале значително. Времето изминато во извршување на операциите insert и update изнесува **842ms** и **1s 63ms**.

```
[2026-05-09 23:19:40] advdb_202526l_prj_cdb_app.public> INSERT INTO ReservationPayment (user_id, reservation_id, employee_id, amount, payment_date, payment_method)
                                                        VALUES (1, 1, 1, 500, CURRENT_DATE, 'Cash')
[2026-05-09 23:19:41] 1 row affected in 842 ms
[2026-05-09 23:19:49] advdb_202526l_prj_cdb_app.public> UPDATE ReservationPayment SET amount = 600 WHERE reservation_id = 1
[2026-05-09 23:19:50] 2 rows affected in 1 s 63 ms
```

Select				
Aggregate		8988	353777.26	349068.1
Unknown (Gather Merge)		35952	353372.8	349068.1
Sort		8988	348090.52	348068.05
Aggregate		8988	345293.81	345158.99
Hash Join		83333	344534.0	200939.0
Full Scan (Seq Scan)	table: reservationpayment;	2500000	103645.0	0.0
Transformation (Hash)		83333	194363.34	194363.34
Hash Join		83333	194363.34	22948.52
Full Scan (Seq Scan)	table: reservation;	2500000	161771.0	0.0
Transformation (Hash)		18945	22711.7	22711.7
Hash Join		18945	22711.7	15.04
Full Scan (Seq Scan)	table: showtime;	568345	20366.45	0.0
Transformation (Hash)		7	14.96	14.96
Hash Join		7	14.96	10.39
Full Scan (Seq Scan)	table: hall;	200	4.0	0.0
Transformation (Hash)		1	10.38	10.38
Full Scan (Seq Scan)	table: cinema;	1	10.38	0.0

10. Нема потреба да се преуреди прашалникот.