Напредни бази на податоци

Фаза 4- Индекси и оптимизација на прашалници

Проект: EventTour

Анастасија Николовска 231507

Теа Русевска 231056

Туана Абуш 231501

View1: Customer Notifications

1. Примарен филтер за погледот vw_customer_notifications ќе биде според customer_id (ID на корисникот/купувачот), а дополнително ќе се користи сортирање според датумот на креирање (created_at).

2. Примарен случај на употреба ќе биде за преглед на центарот за известувања кај корисникот. За овој поглед ни се важни перформансите, бидејќи бавното вчитување на известувањата директно влијае на корисничкото искуство и ефикасноста на апликацијата.

3. Иницијалното време за извршување на погледот е 2 s 665 ms.

```
[2026-05-05 20:53:46] advdb_202526l_prj_even...r.public> SELECT *
                                                         FROM vw_customer_notifications
                                                         WHERE customer_id = 2
[2026-05-05 20:53:49] 500 rows retrieved starting from 1 in 2 s 665 ms (execution: 2 s 144 ms, fetching: 521 ms)
```

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

4. Најбавните операции се full scan на табела notification

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Unknown (Gather Merge)		680	257543.01	257463.67
Sort		340	256464.49	256463.64
Full Scan (Seq Scan)	table: notification;	340	256449.35	0

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

```
[2026-05-05 21:18:05] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) INSERT INTO notification (customer_id, title, message, created_at, is_read, event_id)
                                                         VALUES (2, 'New Update', 'Your ticket has been confirmed.', NOW(), false, 1)
[2026-05-05 21:18:05] completed in 506 ms
```

```
[2026-05-05 21:21:32] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) UPDATE notification
                                                         SET is_read = true
                                                         WHERE customer_id = 2
[2026-05-05 21:21:32] completed in 206 ms
```

5. Времето изминато во извршување на query-то со индекси изнесува 203ms, и тоа е прифатливо време.

```
CREATE INDEX idx_notification_customer_id_created
ON notification (customer_id, created_at DESC);
```

```
[2026-05-05 22:03:10] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) SELECT *
                                                         FROM vw_customer_notifications
                                                         WHERE customer_id = 2
[2026-05-05 22:03:10] completed in 203 ms
```

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

```
[2026-05-05 22:10:13] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) INSERT INTO notification (customer_id, title, message, created_at, is_read, event_id)
                                                         VALUES (2, 'Optimization Test', 'Checking insert speed after indexing.', NOW(), false, 1)
[2026-05-05 22:10:14] completed in 209 ms
```

```
[2026-05-05 22:12:47] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) UPDATE notification
                                                         SET is_read = true
                                                         WHERE customer_id = 2
[2026-05-05 22:12:47] completed in 81 ms
```

View2: Customer Purchases

1. Примарен филтер за погледот vw_customer_purchases ќе биде според customer_id (ID на купувачот), а дополнително може да се користи сортирање според датумот на плаќање (payment_date).

2. Примарен случај на употреба е за преглед на историјата на нарачки и активни билети кај корисникот ("My Tickets"). Бидејќи овој поглед прави спојување (JOIN) на 6 табели, оптимизацијата е клучна за да се избегне прекумерно оптоварување на базата.

3. Иницијалното време за извршување на погледот е скоро 3 секунди. Ова не е прифатливо време за апликацијата па затоа пристапуваме кон индексирање

```
[2026-05-05 22:27:26] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) SELECT *
                                                         FROM vw_customer_purchases
                                                         WHERE customer_id = 10
[2026-05-05 22:27:28] completed in 2 s 279 ms
```

4. За овој поглед, базата врши неколку скапи операции бидејќи се вклучени 6 табели:
   Sequential Scan на табелата ticket: Бидејќи нема индекс на customer_id, базата ја скенира целата табела со билети за да ги најде оние на корисникот.
   Nested Loop Joins: Овие се многу бавни кога се комбинираат податоци од event, venue, seat и payment без претходно индексирани надворешни клучеви.

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Hash Join		400	309621.25	1008.93
Nested Loops (Nested Loo		400	309612.68	1001.43
Nested Loops (Nested L		400	307158.71	1001.15
Index Scan	table: customer; index: customer_pkey;	1	8.3	0.29
Unknown (Gather)		400	307146.4	1000.86
Nested Loops (Ne		167	306106.4	0.86
Nested Loops		167	304700.26	0.43
Full Scan (S	table: ticket;	167	303316.25	0
Index Scan	table: seat; index: seat_pkey;	1	8.29	0.43
Index Scan	table: payment; index: idx_payment_id_status;	1	8.42	0.43
Unknown (Memoize)		1	7.19	0.29
Index Scan	table: event; index: event_pkey;	1	7.18	0.28
Transformation (Hash)		200	5	5
Full Scan (Seq Scan)	table: venue;	200	5	0

5. Времето изминато во извршување на query-то со индекси изнесува 440 ms, и тоа е прифатливо време.

```
[2026-05-05 22:50:21] advdb_202526l_prj_even...r.public> CREATE INDEX idx_ticket_customer_id ON ticket (customer_id)
[2026-05-05 22:53:31] completed in 3 m 10 s 157 ms
```

```
[2026-05-05 22:55:08] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) SELECT *
                                                         FROM vw_customer_purchases
                                                         WHERE customer_id = 10
[2026-05-05 22:55:09] completed in 440 ms
```

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Nested Loops (Nested Loop)		400	8320.64	69.18
Nested Loops (Nested Loop)		400	4952.64	68.75
Hash Join		400	1637.64	68.32
Hash Join		400	1629.06	60.82
Nested Loops (Nestec		400	1575.01	7.82
Index Scan	table: customer; index: customer_pkey;	1	8.3	0.29
Bitmap Index Scan	table: ticket;	400	1562.71	7.53
Bitmap Index Sc	index: idx_ticket_customer_id;	400	7.43	0
Transformation (Hash)		1200	38	38
Full Scan (Seq Sca	table: event;	1200	38	0
Transformation (Hash)		200	5	5
Full Scan (Seq Scan)	table: venue;	200	5	0
Index Scan	table: seat; index: seat_pkey;	1	8.29	0.43
Index Scan	table: payment; index: idx_payment_id_status;	1	8.42	0.43

View 3: Event Sponsors

1. Примарен филтер за погледот vw_event_sponsors ќе биде според event_id (ID на настанот), а дополнително може да се користи пребарување според името на спонзорот (sponsor_name).

2. Примарен случај на употреба е за административен преглед на финансиската поддршка за конкретен настан. Овој поглед е клучен за организаторите за да имаат увид во склучените договори и износите на спонзорствата.

3. Иницијалното време за извршување на погледот е 762ms. Ова е прифатливо време за апликацијата.

```
[2026-05-05 23:04:34] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) SELECT *
                                                         FROM vw_event_sponsors
                                                         WHERE event_id = 5
[2026-05-05 23:04:35] completed in 762 ms
```

4. Најбавните операции се full scan table certificate и тие не можат толку многу да се подобрат со индекси.

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Hash Join		72	1858.35	1795.05
Nested Loops (Nested Loop)		72	1856.89	1793.89
Index Scan	table: event; index: event_pkey;	1	8.29	0.28
Hash Join		72	1847.87	1793.61
Full Scan (Seq Scan)	table: sponsor;	2000	49	0
Transformation (Hash)		72	1792.71	1792.71
Full Scan (Seq Scan	table: event_sponsorship;	72	1792.71	0
Transformation (Hash)		7	1.07	1.07
Full Scan (Seq Scan)	table: sponsor_type;	7	1.07	0

Времето изминато во извршување на операциите insert и update изнесува

```
[2026-05-05 23:09:02] advdb_202526l_prj_even...r.public> INSERT INTO event_sponsorship (event_id, sponsor_id, sponsorship_amount, contract_date)
                                                         VALUES (5, 1, 5000.00, CURRENT_DATE)
[2026-05-05 23:09:04] 1 row affected in 1 s 753 ms
```

5. Нема потреба да се преуреди прашалникот.

View4: Admin Roles

1. Примарниот филтер за погледот vw_admin_roles е според презимето (last_name), а дополнително се користи сортирање според името (first_name).

2. Времето на извршување без индекси изнесува **69 ms**.

```
[2026-05-07 23:13:07] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) select * from vw_admin_roles
                                                         where last_name like 'S%'
                                                         order by first_name
[2026-05-07 23:13:07] completed in 69 ms
```

3. Иако ова време е под границата од 2 секунди, планот на извршување покажува Seq Scan (целосно скенирање) на табелата и посебна Sort операција. Со зголемување на бројот на записи до милион ова време значително ќе се зголеми.

Најбавните операции се Full Scan (Seq Scan) на табелите ticket_admin и event_admin, како и операцијата Sort за името. Ова би претставувало сериозен проблем со перформансите доколку бројот на администратори порасне на милион редици.

4. Времето изминато во извршување на операциите insert и update изнесува:

```
[2026-05-07 23:30:30] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) INSERT INTO admin (email, first_name, last_name, password_hash)
                                                         VALUES ('test_admin@eventtour.com', 'Test', 'Optimization', 'hashed_pass_123')
[2026-05-07 23:30:31] completed in 1 s 584 ms
```

```
[2026-05-07 23:33:42] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) UPDATE admin
                                                         SET first_name = 'Updated_Test'
                                                         WHERE email = 'test_admin@eventtour.com'
[2026-05-07 23:33:43] completed in 414 ms
```

5. Времето изминато во извршување на query-то со индекси изнесува 47ms тоа е прифатливо време.

```
[2026-05-07 23:39:28] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) SELECT * FROM vw_admin_roles
                                                         WHERE last_name LIKE 'T%'
                                                         ORDER BY first_name
[2026-05-07 23:39:28] completed in 47 ms
```

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

```
[2026-05-07 23:42:19] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) INSERT INTO admin (email, first_name, last_name, password_hash)
                                                         VALUES ('test_admin2@eventtour.com', 'Simeon', 'Sulev', 'hash456')
[2026-05-07 23:42:19] completed in 50 ms
```

```
[2026-05-07 23:43:11] advdb_202526l_prj_even...r.public> EXPLAIN (FORMAT JSON) UPDATE admin
                                                         SET first_name = 'Anastasija'
                                                         WHERE email = 'ane_admin@eventtour.com'
[2026-05-07 23:43:12] completed in 38 ms
```

View5: Event Details

1. Примарен филтер за погледот vw_event_details е според ID-то на настанот (event_id).

2. Примарен случај на употреба е за прикажување на деталните информации за еден конкретен настан на корисникот (вклучувајќи ја точната локација, категоријата и сликата). За овој поглед ни се исклучително важни перформансите, бидејќи најчесто ќе се повикува во апликацијата и бавното вчитување директно го нарушува корисничкото искуство.

3. Иницијалното време за извршување на погледот е **2 s 47 ms**. Според барањата за проектот, ова не е прифатливо време (надминува 2 секунди), па затоа задолжително пристапуваме кон индексирање.

```
[2026-05-09 00:12:42] advdb_202526l_prj_even...r.public> SELECT * FROM vw_event_details
                                                         WHERE event_id = 89
[2026-05-09 00:12:44] 1 row retrieved starting from 1 in 2 s 47 ms (execution: 394 ms, fetching: 1 s 653 ms)
```

4. Најбавните операции се *Nested Loop Joins* и *Full Scan* на табелите бидејќи недостасуваат индекси на надворешните клучеви кои се користат за поврзување на податоците, како и за подпрашалникот за медиуми.

Output | Query Plan | advdb_202526l_prj_ev...blic.vw_event_details

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Nested Loops (Nested Loop)		1	21.28	14.29
Nested Loops (Nested Loop		1	21.12	14.15
Hash Join		1	15.43	13.86
Full Scan (Seq Scan)	table: category;	41	1.41	0
Transformation (Hash)		1	13.84	13.84
Hash Join		1	13.84	8.31
Full Scan (Seq S	table: venue;	200	5	0
Transformation (Ha		1	8.29	8.29
Index Scan	table: city; index: city_pkey;	1	5.69	0.29
Index Scan	table: country; index: country_pkey;	1	0.16	0.14

Property	Value
Operation	Nested Loop
Rows	1
Total Cost	21.12
Startup Cost	14.15
Parent Relationship	Outer
Parallel Aware	false
Async Capable	false
Join Type	Inner
Plan Width	107
Inner Unique	true

5. Времето изминато во извршување на операциите insert и update изнесува

```
[2026-05-09 00:23:06] advdb_202526l_prj_even...r.public> INSERT INTO event_media (event_id, url)
                                                         VALUES (89, 'https://example.com/test_image.jpg')
[2026-05-09 00:23:07] 1 row affected in 325 ms
```

```
[2026-05-09 00:24:11] advdb_202526l_prj_even...r.public> UPDATE event
                                                         SET status = 'ONGOING'
                                                         WHERE event_id = 89
[2026-05-09 00:24:12] 1 row affected in 1 s 18 ms
```

6. Времето изминато во извршување на query-то со индекси изнесува 537ms и тоа е прифатливо време.

```
[2026-05-09 00:26:29] advdb_202526l_prj_even...r.public> SELECT * FROM vw_event_details
                                                         WHERE event_id = 89
[2026-05-09 00:26:29] 1 row retrieved starting from 1 in 537 ms (execution: 80 ms, fetching: 457 ms)
```

Output | Query Plan | advdb_202526l_prj_ev...blic.vw_event_details

Operation	Params	Rows	Total Cost	Startup Cost
Select				
Nested Loops (Nested Loop)		1	21.28	14.29
Nested Loops (Nested Loop		1	21.12	14.15
Hash Join		1	15.43	13.86
Full Scan (Seq Scan)	table: category;	41	1.41	0
Transformation (Hash)		1	13.84	13.84
Hash Join		1	13.84	8.31
Full Scan (Seq S	table: venue;	200	5	0
Transformation (Ha		1	8.29	8.29
Index Scan	table: city; index: city_pkey;	1	5.69	0.29
Index Scan	table: country; index: country_pkey;	1	0.16	0.14

Property	Value
Operation	index scan
Table	city
Index	city_pkey
Rows	1
Total Cost	5.69
Startup Cost	0.29
Parent Relationship	Inner
Parallel Aware	false
Async Capable	false
Scan Direction	Forward
Alias	c
Plan Width	26
Index Cond	(city_id = v.city_id)

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

```
[2026-05-09 00:32:01] advdb_202526l_prj_even...r.public> INSERT INTO event_media (event_id, url)
                                                         VALUES (91, 'https://example2.com/test2_image2.jpg')
[2026-05-09 00:32:01] 1 row affected in 308 ms
```

```
[2026-05-09 00:31:18] advdb_202526l_prj_even...r.public> UPDATE event
                                                         SET status = 'ONGOING'
                                                         WHERE event_id = 90
[2026-05-09 00:31:18] 1 row affected in 59 ms
```

View 6: Customer Profile Stats

1. Примарен филтер за овој поглед ќе биде customer_id. Погледот се користи во табот "Мој профил" каде корисникот гледа аналитика за својата активност (вкупно купени билети, непрочитани известувања, лојален статус).

2. Бидејќи погледот се вчитува при секое отворање на профилот, перформансите се клучни. Сепак, прашалникот е аналитички и содржи повеќе поврзани подпрашалници (correlated subqueries) и агрегации (COUNT, MAX) кои се пресметуваат за секој корисник одделно преку масивни табели како ticket, notification и payment.

3. Иницијалното време за извршување на стандардниот поглед е 5s 767ms. Поради природата на агрегациите, додавањето на обични индекси на основните табели нема значително да го подобри времето на извршување при пребарување низ милиони редици.

```
[2026-05-09 00:53:21] advdb_202526l_prj_even...r.public> SELECT * FROM vw_customer_profile_stats WHERE customer_id = 565
[2026-05-09 00:53:27] 1 row retrieved starting from 1 in 5 s 767 ms (execution: 5 s 398 ms, fetching: 369 ms)
```

4. Оптимизација (Материјализиран поглед): За да се реши овој проблем, наместо стандарден поглед, креиран е Материјализиран поглед (Materialized View). Ова овозможува сите тешки пресметки да се извршат однапред и резултатот да се зачува физички. Дополнително, врз самиот материјализиран поглед креиран е Unique Index на customer_id (idx_mv_customer_profile_id).

5. Времето на извршување по креирањето на материјализираниот поглед изнесува 792 ms, што е драстично подобрување и овозможува инстант вчитување на профилот.

```
[2026-05-10 23:06:03] advdb_202526l_prj_even...r.public> SELECT * FROM mv_customer_profile_stats WHERE customer_id = 15
[2026-05-10 23:06:04] 1 row retrieved starting from 1 in 792 ms (execution: 111 ms, fetching: 681 ms)
```

6. *Забелешка:* Бидејќи податоците во материјализираниот поглед се физички зачувани, времето на извршување на операциите INSERT и UPDATE во основните табели останува исто (нема забавување). За да се прикажат најновите податоци, потребно е само повремено освежување на погледот преку командата REFRESH MATERIALIZED VIEW.

View 7: Event Revenue

1. Примарен филтер за погледот vw_event_revenue ќе биде според event_id (ID на настанот) или неговото име (name).

2. Овој поглед е од суштинско значење за менаџментот и организаторите, бидејќи овозможува брз увид во финансиската успешност на секој настан преку пресметка на вкупниот приход од успешно продадените билети.

3. Иницијалното време за извршување на стандардниот поглед е . Бидејќи времето надминува 2 секунди и вклучува сложени агрегации врз милиони редици во табелата ticket, неопходна е оптимизација.

4. Најголем товар врз перформансите прават операциите Hash Aggregate и Nested Loop Joins. Поради големиот волумен на податоци во табелите за билети и плаќања, базата троши многу ресурси за пресметување на сумата во реално време.1
5. За овој поглед се применува стратегија на Materialized View (mv_event_revenue). Со физичко зачувување на резултатите од агрегацијата, се елиминира потребата од повторно пресметување при секој пристап. Дополнително е додаден Unique Index на event_id во самиот материјализиран поглед.

6. Заклучок: Со користење на материјализиран поглед, времето на извршување е драстично намалено, овозможувајќи им на администраторите инстант пристап до извештаите за приходи.

7. Бидејќи ова е материјализиран поглед, тој не се ажурира автоматски кога ќе се продаде нов билет. Во реална апликација, би користеле команда за освежување:

-- Оваа команда се активира периодично или преку тригер по завршување на продажба
REFRESH MATERIALIZED VIEW mv_event_revenue;

View 8: Event Archive

1. Примарен филтер за овој поглед ќе биде според ID на настанот (event_id), а дополнително може да се пребарува според името на настанот или категоријата.

2. Овој поглед е наменет за приказ на архивирани (завршени) настани заедно со нивните оценки и вкупен број на рецензии. Ова е клучно за корисниците кои сакаат да ја видат историјата и успешноста на минатите настани.

3. Бидејќи времето надминува 2 секунди и вклучува сложени агрегации врз табелата со рецензии која содржи голем број записи, неопходна е оптимизација овде индекси не помагаат.

4. Проблематични операции: Најголем товар врз перформансите прават операциите Hash Aggregate и Sequential Scan врз табелата review. Базата троши многу ресурси за пресметување на просечната оценка во реално време.

5. Се применува стратегија на Materialized View (mv_event_archive). Со физичко зачувување на пресметаните резултати, се елиминира потребата од повторно пресметување при секој пристап. Дополнително е додаден уникатен индекс на event_id во самиот материјализиран поглед.

6. Заклучок: Со користење на материјализиран поглед, времето на извршување е драстично намалено, овозможувајќи им на корисниците инстант пристап до архивата со оценки.

View 9: Event Ticket Master Report

За овој поглед ситуацијата е специфична бидејќи овој поглед користи агрегатни функции со филтри (FILTER (WHERE ...)) врз колоната status.

1. Зошто стандардното индексирање не помага?

Иако вообичаено додаваме индекси на колони кои се користат во WHERE клаузулата, кај овој поглед имаме неколку пречки:

- Агрегација на цела табела: Кверито мора да ги преброи *сите* редови во табелата ticket за секој настан за да ги пресмета вредностите (available, sold, reserved). Дури и да постои индекс на status, базата сепак мора да направи Index Scan или Bitmap Scan на огромен дел од индексот, што кај табели со милиони редови не е значително побрзо од Sequential Scan.
- Мултипликативни филтри: Бидејќи имаме 4 различни COUNT функции со 4 различни филтри, базата најчесто одлучува дека е поефикасно да ја прочита целата табела еднаш (Sequential Scan) и да ги распредели редовите во меморијата, отколку 4 пати да скока низ индекси за различни статуси.
- Пресметковни колони: Колоната sales_occupancy_percent е пресметковна и зависи од резултатите на агрегацијата. Индексот не може да ги „запамети“ овие математички операции.

2. Каде е Bottleneck?

Главниот проблем со перформансите кај ова квери се:

1. **Hash Aggregate:** Базата мора да ги групира милионите записи од ticket по event_id. Ова бара многу меморија (RAM) и процесорска моќ.
2. **Large Joins:** Поврзувањето (Join) на event со ticket (каде има милиони записи) создава огромен сет на податоци кој мора да се процесира пред да почне броењето.
3. **Real-time Calculation:** Секој пат кога менаџерот ќе го отвори овој извештај, базата одново ги пресметува процентите и сумите за секој настан, што е непотребно за податоци кои не се менуваат секоја милисекунда.

За View 9 класичното индексирање не нуди подобрување бидејќи базата мора да врши агрегација врз целата содржина на табелата ticket. Иако Materialized View е прифатливо решение за овој случај (како што беше применето кај претходните аналитички погледи), овде го задржуваме стандардниот поглед за да ја прикажеме разликата во перформанси помеѓу пресметки во реално време и оптимизирани физички структури.