Напредни бази на податоци

Фаза 3: Оптимизација на прашалници и погледи

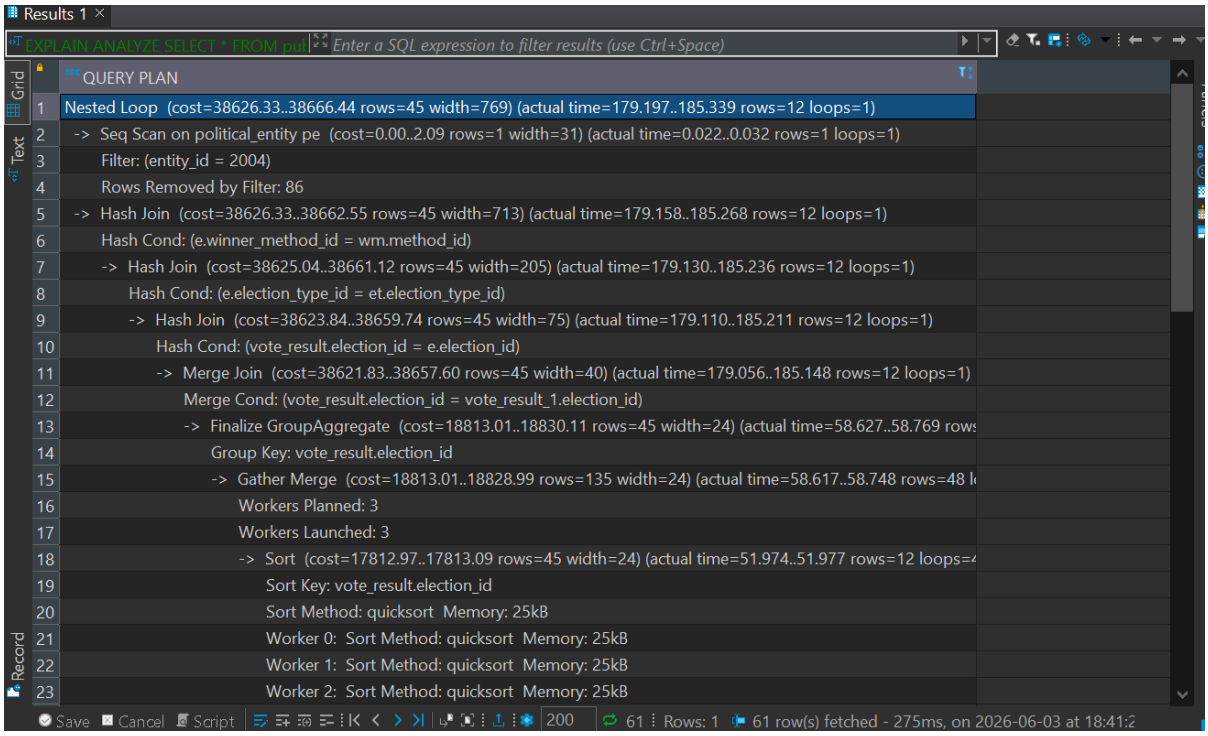
Проект: Платформа за евиденција на избори EDB

1. vw_election_results

1. Примарен филтер за погледот vw_election_results е идентификаторот на политичкиот субјект (entity_id), бидејќи корисниците најчесто ја анализираат распределбата на гласовите и успехот на конкретна партија или коалиција низ изборите и вкупни гласови. Дополнително, погледот може да се филтрира и според идентификаторот на изборите (election_id) со цел добивање на подетални статистики за конкретен изборен циклус.
2. Примарната намена на погледот vw_election_results е детална анализа на вкупниот број освоени гласови по политички субјекти за дадени избори. Со негова помош може да се анализира колку гласачи гласале за одредена партија, да се споредува успехот на партиите во различни изборни години и да се визуелизира победничкиот метод на корисничкиот интерфејс. Бидејќи овој поглед служи како основа за финалните аналитички извештаи и графикони, неговите перформанси се од голема важност.
3. Иницијалното време за извршување изнесува 581ms , бидејќи базата врши агрегација и пребарување на историските податоци за партијата низ сите изборни циклуси.

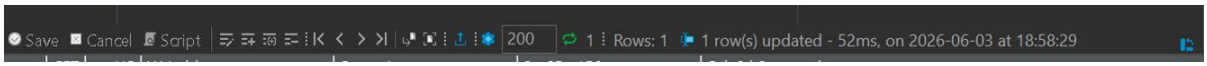
```
EXPLAIN ANALYZE

SELECT * FROM public.vw_election_results

WHERE entity_id = 2004;
```

4. Анализата покажува дека базата користи повеќекратни Hash Join и Merge Join операции за поврзување на табелите, при што извршува секвенцијално скенирање (Seq Scan) врз political_entity и вклучува три паралелни работници за сортирање во меморија. Бидејќи времето е задоволително, се предлага задржување на моменталната структура без дополнителни индекси.

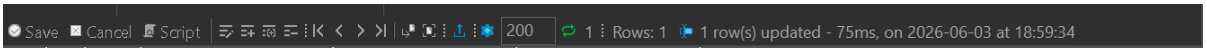


5. Времето изминато во извршување на операциите insert и update изнесува:

Insert : 219 ms



Update: 75ms



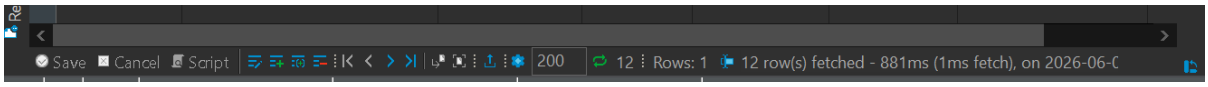
2. party_performance_over_time

1. Примарен филтер за погледот vw_party_performance_over_time е идентификаторот на политичкиот субјект (entity_id), бидејќи корисниците најчесто го анализираат историскиот успех и трендот на една конкретна партија. Дополнително, погледот може да се филтрира и според датумот на изборите (election_date) за да се добијат подетални хронолошки статистики.
2. Примарната намена на погледот vw_party_performance_over_time е анализа на растот или падот на рејтингот на партиите низ различни изборни циклуси. Со

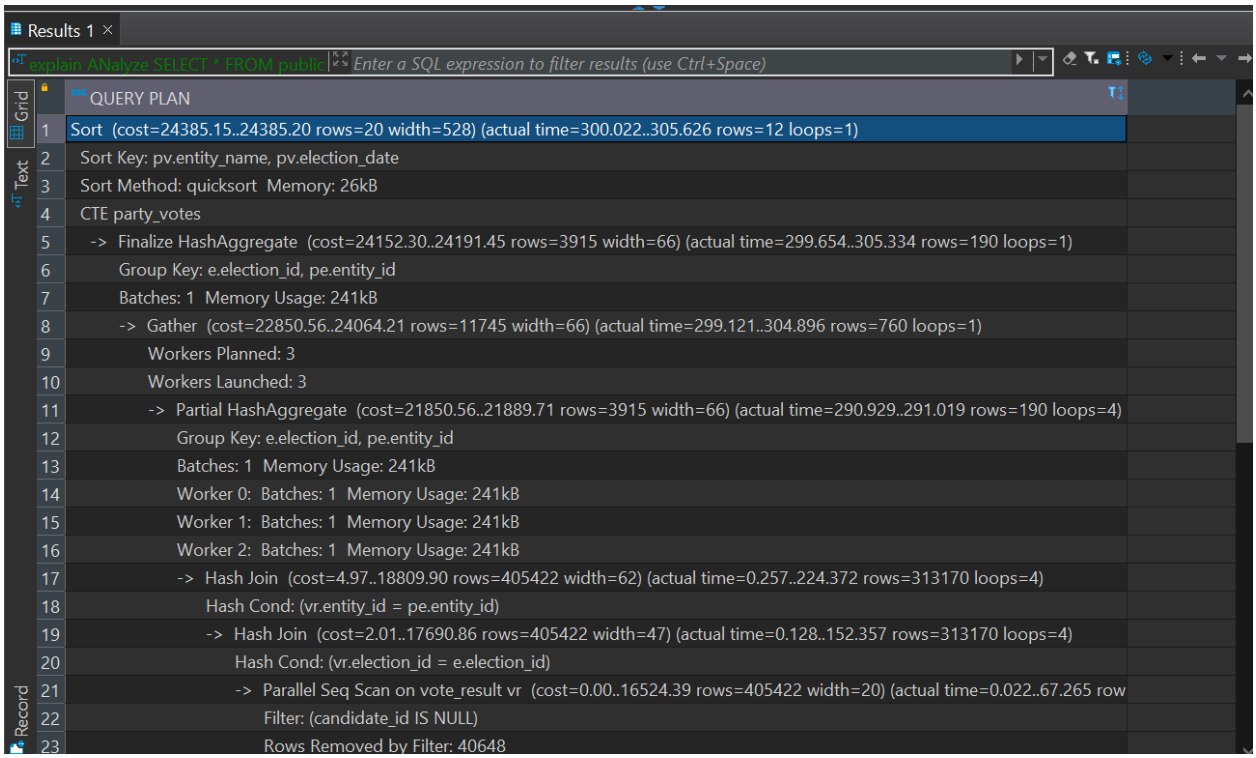
негова помош се генерираат аналитички графикони кои го прикажуваат историскиот перформанс на политичките субјекти.

3. Иницијално време за извршување е 880 ms

```
EXPLAIN ANALYZE

SELECT * FROM public.vw_party_performance_over_time

WHERE entity_id = 2004;
```

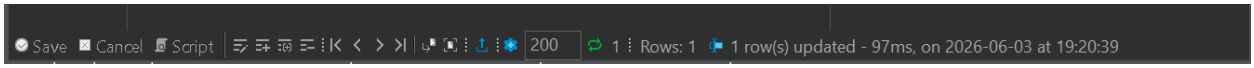


4. За оптимизација, базата вклучува три паралелни работници и врши финално сортирање во меморија преку quicksort. Бидејќи вкупното време од околу 881ms е сосема задоволително за историска аналитика, се предлага задржување на моменталната структура без воведување нови индекси.

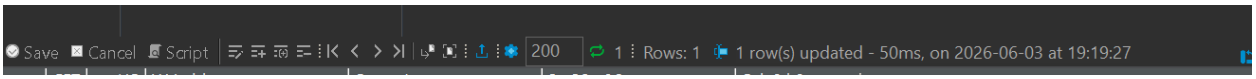


5. Времето изминато во извршување на операциите insert и update

Insert:

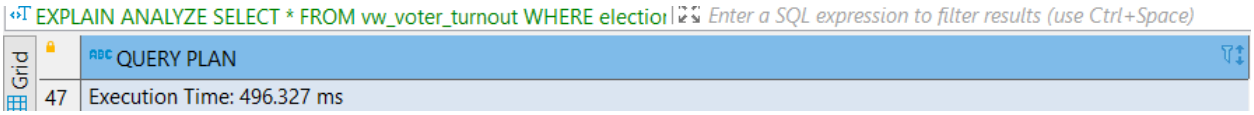


Update:



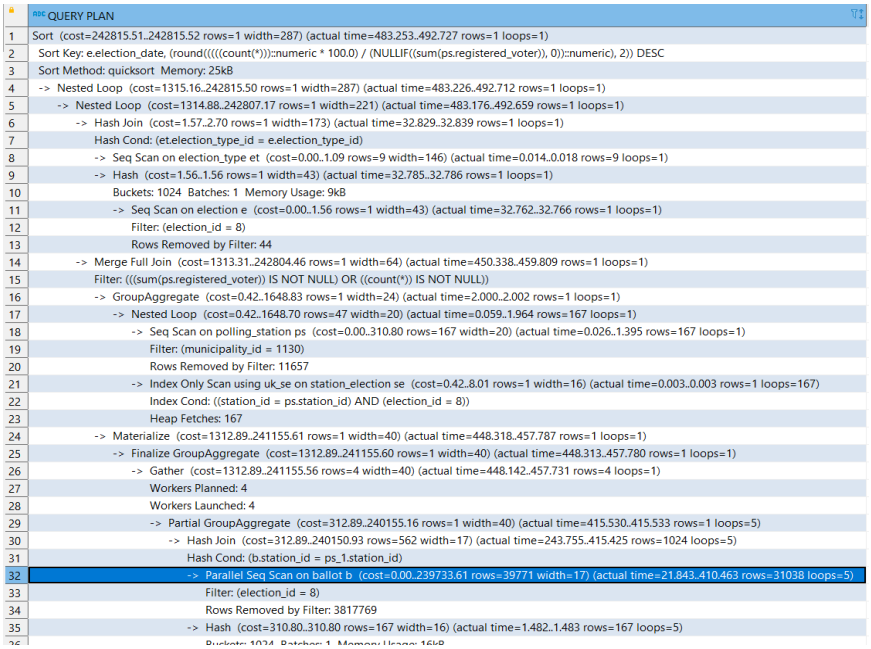
3.vw_voter_turnout

1. Примарен филтер за погледот vw_voter_turnout е идентификаторот на изборите (election_id), бидејќи корисниците најчесто ја анализираат излезноста на гласачите за конкретни избори. Дополнително, погледот може да се филтрира и според општина или регион со цел добивање на подетални статистики за одредена географска област.
2. Примарната намена на погледот vw_voter_turnout е анализа на излезноста на гласачите на различни изборни циклуси. Со негова помош може да се споредува бројот на регистрирани гласачи, бројот на гласале гласачи и процентот на излезност по избори, региони и општини. Бидејќи овој поглед е наменет за статистички анализи и извештаи, неговите перформанси се од голема важност.
3. Иницијално време за извршување на погледот е 496.327ms.



Ова време е релативно високо за аналитички поглед и затоа се разгледува можност за оптимизација преку индексирање.

4. Анализата на планот за извршување покажува дека најголем дел од времето се троши на операцијата **Parallel Seq Scan** врз табелата ballot, при што се пребаруваат голем број записи за да се пронајдат оние што припаѓаат на конкретните избори. Поради тоа се предлага креирање индекс врз колоната election_id во табелата ballot.



Времето изминато во извршување на операциите insert и update изнесува:

	QUERY PLAN
1	Update on ballot (cost=0.44..8.46 rows=0 width=0) (actual time=17.953..17.954 rows=0 loops=1)
2	-> Index Scan using ballot_pkey on ballot (cost=0.44..8.46 rows=1 width=7) (actual time=17.727..17.730 rows=1 loops=1)
3	Index Cond: (ballot_id = 79382834)
4	Planning Time: 0.104 ms
5	Execution Time: 17.992 ms

	QUERY PLAN
1	Insert on ballot (cost=0.00..0.02 rows=0 width=0) (actual time=0.817..0.818 rows=0 loops=1)
2	-> Result (cost=0.00..0.02 rows=1 width=49) (actual time=0.023..0.023 rows=1 loops=1)
3	Planning Time: 0.050 ms
4	Trigger for constraint fk_ballot_election: time=0.210 calls=1
5	Trigger for constraint fk_ballot_station: time=0.188 calls=1
6	Trigger for constraint fk_ballot_entity: time=0.494 calls=1
7	Trigger for constraint fk_ballot_candidate: time=0.053 calls=1
8	Trigger trg_after_insert_ballot_live_count: time=17.456 calls=1
9	Trigger trg_validate_ballot: time=0.698 calls=1
10	Execution Time: 19.242 ms

5. Времето на извршување на погледот по индексирање изнесува 101.735ms. Анализата на новиот план за извршување покажува дека наместо Parallel Seq Scan врз табелата ballot, базата користи Bitmap Index Scan преку индексот idx_ballot_election_id, со што значително се намалува бројот на обработени записи.

```
CREATE INDEX idx_ballot_election_id ON ballot(election_id);
```

sults 1 × Execution plan - 1

PLAIN ANALYZE SELECT * FROM vw_voter_turnout WHERE electio Enter a SQL expression to filter results (use Ctrl+Space)

	QUERY PLAN
	Rows Removed by Filter: 11657
	-> Index Scan using region_pkey on region rg (cost=0.28..8.29 rows=1 width=18) (actual time=0.021..0.022 rows=1 loop
	Index Cond: (region_id = 1130)
	Planning Time: 1.119 ms
	JIT:
	Functions: 117
	Options: Inlining false, Optimization false, Expressions true, Deforming true
	Timing: Generation 8.704 ms (Deform 4.032 ms), Inlining 0.000 ms, Optimization 3.668 ms, Emission 84.196 ms, Total 96.569 r
	Execution Time: 101.735 ms

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

QUERY PLAN
Insert on ballot (cost=0.00..0.02 rows=0 width=0) (actual time=0.980..0.980 rows=0 loops=1)
-> Result (cost=0.00..0.02 rows=1 width=49) (actual time=0.031..0.032 rows=1 loops=1)
Planning Time: 0.059 ms
Trigger for constraint fk_ballot_election: time=0.141 calls=1
Trigger for constraint fk_ballot_station: time=0.108 calls=1
Trigger for constraint fk_ballot_entity: time=0.126 calls=1
Trigger for constraint fk_ballot_candidate: time=0.006 calls=1
Trigger trg_after_insert_ballot_live_count: time=0.350 calls=1
Trigger trg_validate_ballot: time=0.748 calls=1
Execution Time: 1.738 ms

	QUERY PLAN
1	Update on ballot (cost=0.44..8.46 rows=0 width=0) (actual time=0.095..0.096 rows=0 loops=1)
2	-> Index Scan using ballot_pkey on ballot (cost=0.44..8.46 rows=1 width=7) (actual time=0.032..0.035 rows=1 loops=1)
3	Index Cond: (ballot_id = 79382834)
4	Planning Time: 0.106 ms
5	Execution Time: 0.165 ms

4. vw_party_demographic_performance

1. Примарен филтер за погледот е election_id, а дополнително може да се филтрира и според политички субјект (entity_id) и избирачко место (station_id).
2. Погледот се користи за анализа на демографската структура на гласачите по политички партии и генерации. Поради големиот број записи во табелите ballot, voter_election, voter и person, перформансите се значајни.
3. Иницијално време за извршување на погледот е 991.327ms.

```
EXPLAIN ANALYZE SELECT * FROM vw_party_demographic_performa   Enter a SQL expression to filter results (use Ctrl+Space)
```

	QUERY PLAN
54	Execution Time: 991.040 ms

4. Најбавната операција е **full scan (Seq Scan)** врз табелата ballot, бидејќи се обработуваат голем број записи (околу 19 милиони). Се забележува дека PostgreSQL претежно се потпира на Parallel Seq Scan, Seq Scan и Hash Join операции. Ова е очекувано бидејќи се обработува голем дел од податоците во табелата ballot, при што оптимизаторот проценил дека секвенцијалното читање е поефикасно. И покрај тоа, времето на извршување изнесува 991.327 ms (помалку од една секунда), што е одличен резултат за аналитички поглед над милионски број записи. Бидејќи погледот нема честа употреба и перформансите се задоволителни, нема потреба од дополнителна оптимизација.

```
EXPLAIN ANALYZE SELECT * FROM vw_party_demographic_performa   Enter a SQL expression to filter results (use Ctrl+Space)
```

	QUERY PLAN
24	-> Hash (cost=1.87..1.87 rows=87 width=23) (actual time=40.488..40.489 rows=87 loops=5)
25	Hash Cond: (ps.municipality_id = r.region_id)
26	Buckets: 1024 Batches: 1 Memory Usage: 14kB
27	-> Seq Scan on political_entity pe (cost=0.00..1.87 rows=87 width=23) (actual time=40.428..40.440 rows=87 loops=5)
28	-> Hash Join (cost=439.59..242663.61 rows=104164 width=66) (actual time=898.393..898.398 rows=0 loops=5)
29	-> Hash (cost=10.64..10.64 rows=564 width=18) (actual time=0.244..0.244 rows=564 loops=5)
30	Hash Cond: (b.station_id = ps.station_id)
31	Buckets: 1024 Batches: 1 Memory Usage: 38kB
32	-> Seq Scan on region r (cost=0.00..10.64 rows=564 width=18) (actual time=0.060..0.133 rows=564 loops=5)
33	-> Hash Left Join (cost=10.55..241961.04 rows=104164 width=51) (actual time=898.392..898.397 rows=0 loops=5)
34	-> Hash (cost=281.24..281.24 rows=11824 width=31) (never executed)
35	Hash Cond: ((b.election_id = ve.election_id) AND (b.station_id = ve.station_id))
36	-> Seq Scan on polling_station ps (cost=0.00..281.24 rows=11824 width=31) (never executed)
37	-> Hash Join (cost=1.57..241170.83 rows=104164 width=55) (actual time=898.391..898.394 rows=0 loops=5)
38	-> Hash (cost=8.96..8.96 rows=1 width=20) (never executed)
39	Hash Cond: (b.election_id = e.election_id)
40	-> Parallel Seq Scan on ballot b (cost=0.00..227706.09 rows=4687366 width=32) (actual time=0.068..597.224 rows=3752676 loops=5)
41	-> Nested Loop (cost=0.87..8.96 rows=1 width=20) (never executed)
42	-> Hash (cost=1.56..1.56 rows=1 width=31) (actual time=0.054..0.055 rows=1 loops=5)
43	Rows Removed by Filter: 96132
44	Filter: is_valid
45	Buckets: 1024 Batches: 1 Memory Usage: 9kB
46	-> Seq Scan on election e (cost=0.00..1.56 rows=1 width=31) (actual time=0.043..0.047 rows=1 loops=5)
47	-> Nested Loop (cost=0.43..8.45 rows=1 width=24) (never executed)
48	-> Index Scan using person_pkey on person p (cost=0.43..0.51 rows=1 width=12) (never executed)
49	Rows Removed by Filter: 44
50	Index Cond: (person_id = v.person_id)
51	Filter: ((name)::text = 'Presidential Election 2024'::text)
52	-> Seq Scan on voter_election ve (cost=0.00..0.00 rows=1 width=24) (never executed)
53	-> Index Scan using voter_pkey on voter v (cost=0.43..8.45 rows=1 width=16) (never executed)
54	Index Cond: (voter_id = ve.voter_id)

Времето изминато во изршување на операциите insert и update изнесува:

```
EXPLAIN ANALYZE UPDATE ballot SET is_valid = false WHERE ballot   Enter a SQL expression to filter results (use Ctrl+Space)
```

	QUERY PLAN
1	Update on ballot (cost=0.44..8.46 rows=0 width=0) (actual time=15.161..15.162 rows=0 loops=1)
2	-> Index Scan using ballot_pkey on ballot (cost=0.44..8.46 rows=1 width=7) (actual time=15.160..15.160 rows=0 loops=1)
3	Index Cond: (ballot_id = 100000)
4	Planning Time: 0.129 ms
5	Execution Time: 15.201 ms

```
EXPLAIN ANALYZE INSERT INTO ballot (election_id, station_id, entity   Enter a SQL expression to fil
```

Grid | Text | Record

	QUERY PLAN
1	Insert on ballot (cost=0.00..0.02 rows=0 width=0) (actual time=0.724..0.724 rows=0 loops=1)
2	-> Result (cost=0.00..0.02 rows=1 width=49) (actual time=0.016..0.016 rows=1 loops=1)
3	Planning Time: 0.054 ms
4	Trigger for constraint fk_ballot_election: time=0.128 calls=1
5	Trigger for constraint fk_ballot_station: time=0.162 calls=1
6	Trigger for constraint fk_ballot_entity: time=0.117 calls=1
7	Trigger for constraint fk_ballot_candidate: time=0.006 calls=1
8	Trigger trg_after_insert_ballot_live_count: time=20.300 calls=1
9	Trigger trg_validate_ballot: time=0.606 calls=1
10	Execution Time: 21.477 ms

5. Нема потреба од дополнителна оптимизација.
6. Време на извршување останува исто.

5. *vw_polling_station_stats*

1. Примарниот филтер за овој поглед е `election_id`, кој заедно со условот `is_valid = true` ја ограничува анализата само на валидни гласови за конкретни избори. Овој филтер овозможува значително намалување на бројот на обработени записи уште на ниво на табелата `ballot`. Со тоа се подобрува ефикасноста на join и агрегациските операции.
2. Погледот се користи за анализа на активноста на гласачките места преку број на гласови по polling station за дадени избори. Тој овозможува споредба на излезноста меѓу различни станици и изборни циклуси. Дополнително, служи како основа за статистички извештаи и перформанс анализи на изборниот процес.
3. Иницијално време за извршување на погледот е 117.591 ms.

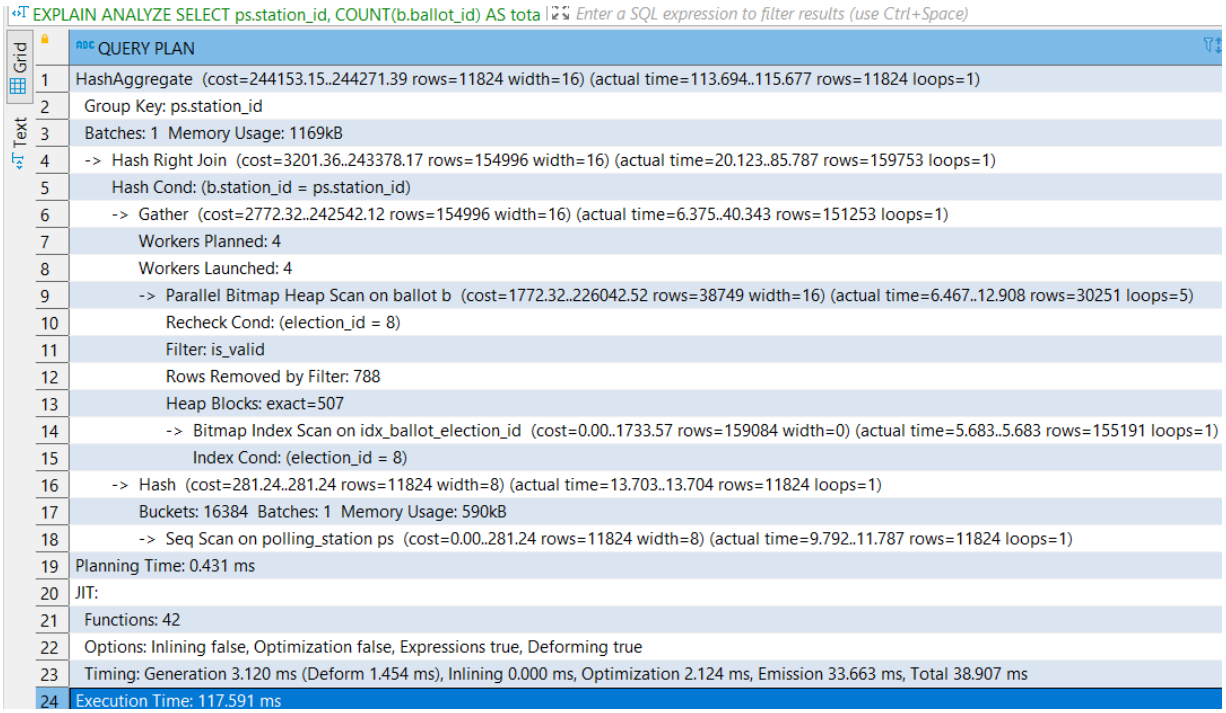
```
EXPLAIN ANALYZE
SELECT ps.station_id,
       COUNT(b.ballot_id) AS total_ballots
FROM polling_station ps
LEFT JOIN ballot b
  ON b.station_id = ps.station_id
 AND b.election_id = 8
 AND b.is_valid = true
GROUP BY ps.station_id;
```

Results 1 | Execution plan - 1

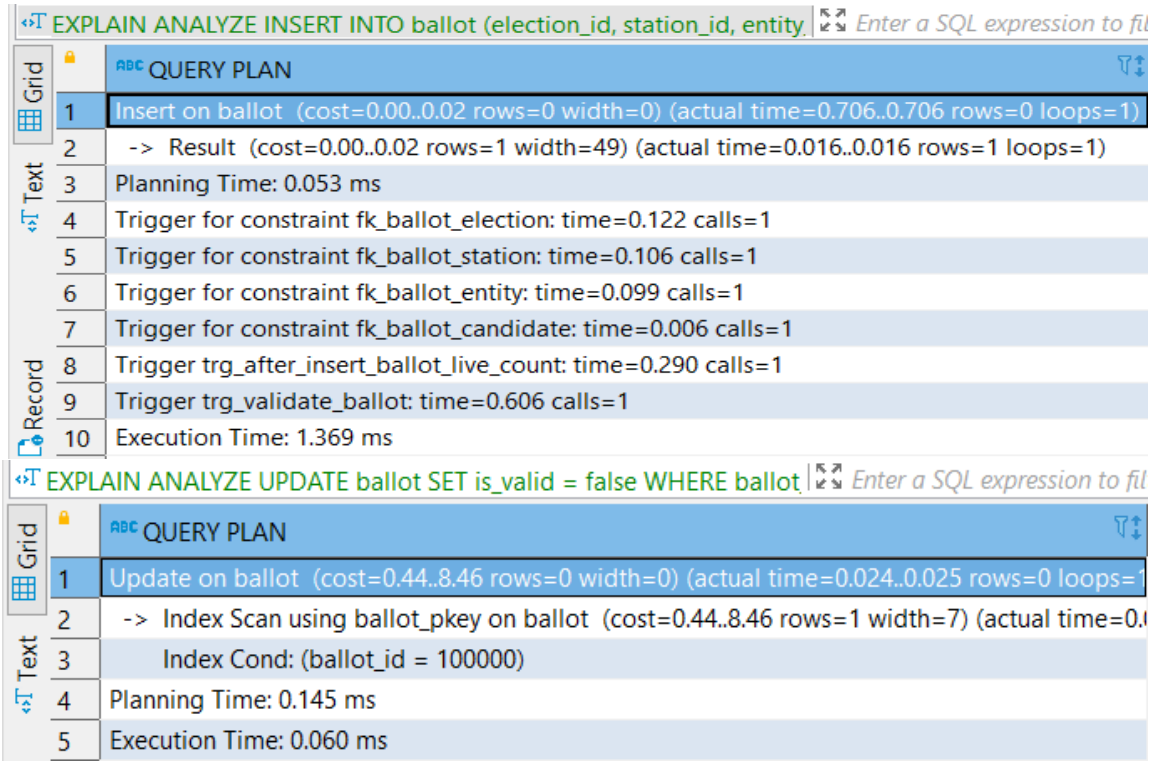
EXPLAIN ANALYZE SELECT ps.station_id, COUNT(b.ballot_id) AS tota | Enter a SQL expression to filter results (use Ctrl+Space)

	QUERY PLAN
24	Execution Time: 117.591 ms

4. Во execution plan се гледа дека веќе креираниот индекс `idx_ballot_election_id` се користи преку `Bitmap Index Scan`, со што се забрзува филтрирањето на податоците. Времето на извршување од ~117ms покажува дека погледот е добро оптимизиран и нема потреба од дополнителни индекси во моменталната верзија.



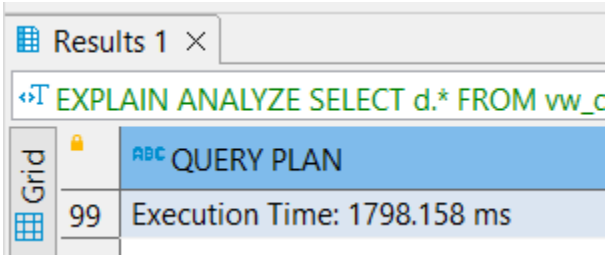
Времето изминато во изршување на операциите insert и update изнесува:



5. Нема потреба од дополнителна оптимизација.
6. Време на извршување останува исто.

6. *vw_dhondt_seat_allocation*

1. Примарен филтер за `vw_dhondt_seat_allocation` е `election_type = 2 (Parliamentary)`, бидејќи D'Hondt методот се применува исклучиво на парламентарни избори. Дополнително може да се филтрира и по `election_id` за анализа на конкретен изборен циклус.
2. Погледот служи за распределба на мандати (seat allocation) по партии користејќи го D'Hondt методот. Се користи за анализа на бројот на освоени места по политички ентитети во парламентарни избори.
3. Иницијално време за извршување на погледот е 1798.158ms.



4. Во извршниот план се користат индексите кои автоматски се креирани преку примарните клучеви и уникатните ограничувања (`PRIMARY KEY` и `UNIQUE`). Најголемо влијание врз времето на извршување имаат `Parallel Seq Scan` врз табелата `vote_result`, како и `HashAggregate`, `WindowAgg` и `Sort` операциите потребни за D'Hondt пресметките.
   Иако времето на извршување изнесува околу **1.8 секунди**, не постои потреба од дополнителна оптимизација бидејќи погледот е наменет за аналитички пресметки кои не се извршуваат често. Поради тоа, моменталните перформанси се прифатливи.

```
EXPLAIN ANALYZE SELECT d.* FROM vw_dhondt_seat_allocation d JC   Enter a SQL expression to filter results (use Ctrl+Space)

QUERY PLAN
Hash Join  (cost=1560212.77..1560496.42 rows=6023 width=168) (actual time=1789.704..1789.942 rows=286 loops=1)
  Hash Cond: (ranked.election_id = e.election_id)
  ->  Sort  (cost=1560210.93..1560347.81 rows=54751 width=168) (actual time=629.775..629.950 rows=286 loops=1)
        Sort Key: ranked.election_id, ranked.district_id, (count(*) FILTER (WHERE (ranked.rn <= ranked.seats_available))) DESC
        Sort Method: quicksort  Memory: 66kB
        CTE district_votes
          ->  HashAggregate  (cost=23907.55..24075.36 rows=16781 width=44) (actual time=601.620..602.838 rows=3391 loops=1)
                Group Key: ed.district_id, vr.entity_id
                Batches: 1  Memory Usage: 1297kB
                ->  Gather  (cost=2484.39..23781.70 rows=16781 width=40) (actual time=85.710..552.937 rows=277081 loops=1)
                      Workers Planned: 3
                      Workers Launched: 3
                      ->  Hash Join  (cost=1484.39..21103.60 rows=5413 width=40) (actual time=434.864..514.695 rows=69270 loops=4)
                            Hash Cond: ((vr.election_id = e_2.election_id) AND (vr.station_id = ps.station_id))
                            ->  Parallel Seq Scan on vote_result vr  (cost=0.00..16524.39 rows=405422 width=28) (actual time=376.311..416.897 rows=313171 loops=4)
                                  Filter: (candidate_id IS NULL)
                                  Rows Removed by Filter: 40648
                            ->  Hash  (cost=1377.78..1377.78 rows=7107 width=44) (actual time=49.025..49.030 rows=52100 loops=4)
                                  Buckets: 65536 (originally 8192)  Batches: 1 (originally 1)  Memory Usage: 4384kB
                                  ->  Hash Join  (cost=981.13..1377.78 rows=7107 width=44) (actual time=22.134..31.332 rows=52100 loops=4)
                                        Hash Cond: (ps.municipality_id = r2.region_id)
                                        ->  Seq Scan on polling_station ps  (cost=0.00..281.24 rows=11824 width=16) (actual time=0.045..1.896 rows=11825 loops=4)
                                        ->  Hash  (cost=976.90..976.90 rows=339 width=44) (actual time=22.048..22.052 rows=1744 loops=4)
                                              Buckets: 2048 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 145kB
                                              ->  Nested Loop  (cost=6.46..976.90 rows=339 width=44) (actual time=0.293..21.367 rows=1744 loops=4)
                                                    ->  Hash Join  (cost=1.62..20.74 rows=105 width=36) (actual time=0.129..0.605 rows=684 loops=4)
                                                          Hash Cond: (ed.election_id = e_2.election_id)
                                                          ->  Seq Scan on electoral_district ed  (cost=0.00..16.41 rows=941 width=28) (actual time=0.034..0.244 rows=941 loops=4)
                                                          ->  Hash  (cost=1.56..1.56 rows=5 width=8) (actual time=0.071..0.071 rows=25 loops=4)
                                                                Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                                                ->  Seq Scan on election e_2  (cost=0.00..1.56 rows=5 width=8) (actual time=0.052..0.058 rows=25 loops=4)
                                                                      Filter: (winner_method_id = 8)
                                                                      Rows Removed by Filter: 20
                                                    ->  Bitmap Heap Scan on region r2  (cost=4.83..9.00 rows=11 width=16) (actual time=0.027..0.027 rows=3 loops=2736)
```

Времето изминато во извршување на операциите insert и update изнесува:

```
EXPLAIN ANALYZE INSERT INTO vote_result ( result_id, election_id, st   Enter a SQL expression to filter r

QUERY PLAN
Insert on vote_result  (cost=0.00..0.01 rows=0 width=0) (actual time=1.544..1.545 rows=0 loops=1)
  ->  Result  (cost=0.00..0.01 rows=1 width=44) (actual time=0.001..0.002 rows=1 loops=1)
Planning Time: 0.047 ms
Trigger for constraint fk_vr_election: time=0.421 calls=1
Trigger for constraint fk_vr_station: time=0.238 calls=1
Trigger for constraint fk_vr_entity: time=0.313 calls=1
Trigger for constraint fk_vr_candidate: time=0.085 calls=1
Trigger for constraint fk_vr_station_election: time=0.529 calls=1
Execution Time: 3.156 ms
```

```
EXPLAIN ANALYZE UPDATE vote_result SET votes = votes + 50 WHEI   Enter a SQL expression to filter re

QUERY PLAN
Update on vote_result  (cost=11.67..819.66 rows=0 width=0) (actual time=0.152..0.154 rows=0 loop
  ->  Bitmap Heap Scan on vote_result  (cost=11.67..819.66 rows=226 width=10) (actual time=0.082
        Recheck Cond: ((election_id = 8) AND (entity_id = 1004) AND (candidate_id IS NULL))
        Heap Blocks: exact=1
        ->  Bitmap Index Scan on uk_vote_result  (cost=0.00..11.62 rows=226 width=0) (actual time=0.
              Index Cond: ((election_id = 8) AND (entity_id = 1004) AND (candidate_id IS NULL))
Planning Time: 0.147 ms
Execution Time: 0.292 ms
```

5. Нема потреба од дополнителна оптимизација.
6. Време на извржување останува исто

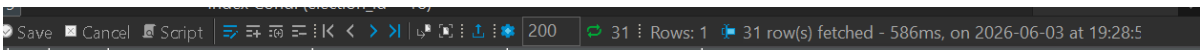
7. *vw_invalid_ballot_analysis*

1. Примарен филтер за погледот vw_invalid_ballot_analysis е идентификаторот на изборите (election_id), бидејќи анализата на неважечките ливчиња секогаш се прави за конкретен изборен циклус. Дополнително, погледот може да се филтрира и според изборна единица или регион (region_id) со цел да се воочат аномалии или специфични трендови на неважечки гласови по географски подрачја.

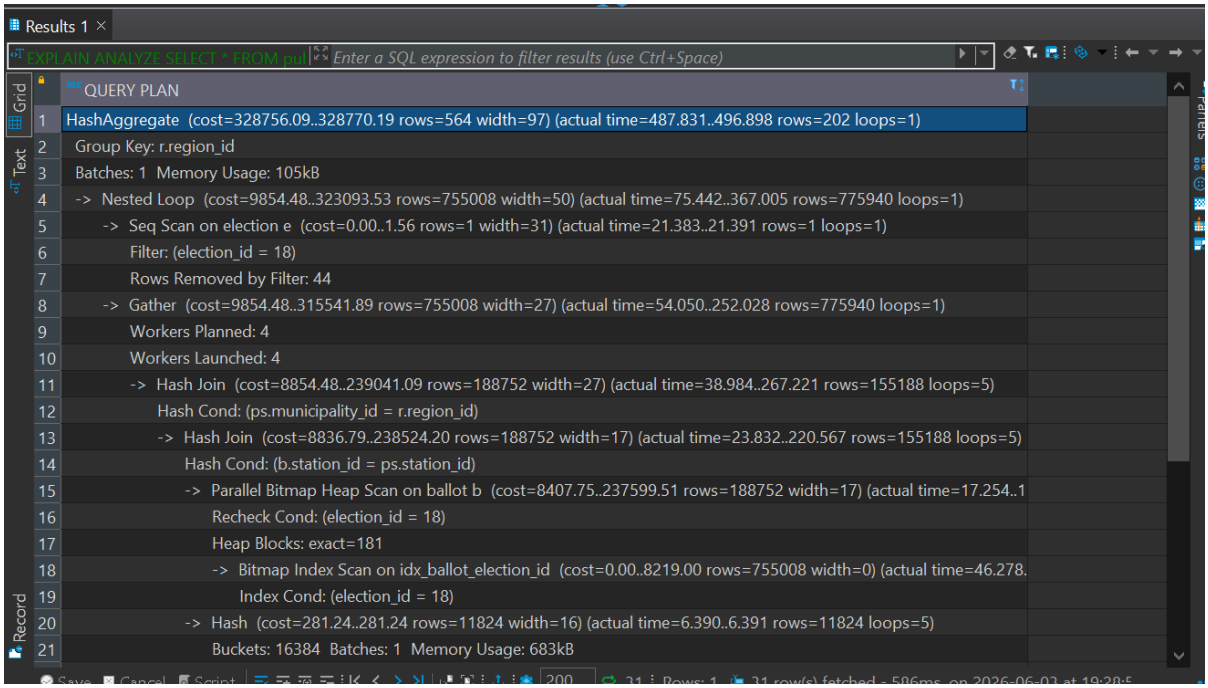
2. Примарната намена на погледот vw_invalid_ballot_analysis е детална статистика и процентуална анализа на неважечките гласачки ливчиња во однос на вкупниот број на ливчиња. Бидејќи погледот врши математички пресметки за проценти над големи табели во реално време, неговите перформанси се клучни.

3. Инцијалното време за извршување на погледот со филтер по изборен циклус изнесува 586 ms (од кои реалното време на извршување во базата е околу 496 ms, а остатокот е за приказ на податоците).

```
EXPLAIN ANALYZE

SELECT * FROM public.vw_invalid_ballot_analysis

WHERE election_id = 18;
```

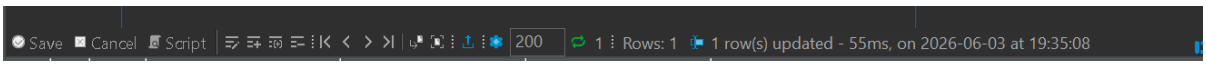


4. Анализата на планот за извршување покажува дека базата успешно го користи постоечкиот индекс врз табелата ballot преку операцијата Bitmap Index Scan (idx_ballot_election_id), со што директно се таргетираат бараните записи. Бидејќи индексот е веќе имплементиран и перформансите се оптимизирани, не се предлагаат дополнителни измени.

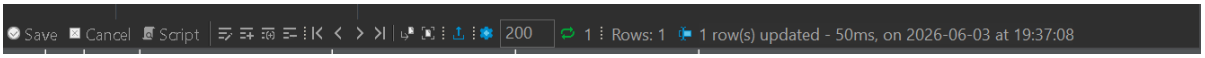


5. Времето изминато во извршување на операциите insert и update врз референтната табела Ballot изнесува:

Insert : 55ms



Update:



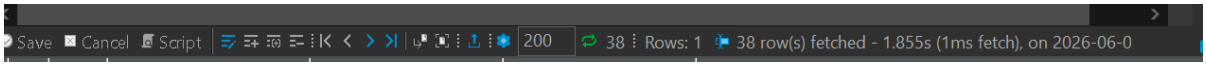
8. *vw_local_election_winners*

1. Примарен филтер за погледот vw_local_election_winners (на пр. 'Local Elections 2013')е името на изборите (election_name), бидејќи анализата на победниците секогаш се врши за одреден изборен циклус .

2. Примарната намена на погледот vw_local_election_winners е брз приказ на победниците (кандидати или партии) по општини за дадените локални избори, заедно со бројот на добиени гласови и нивниот статус.

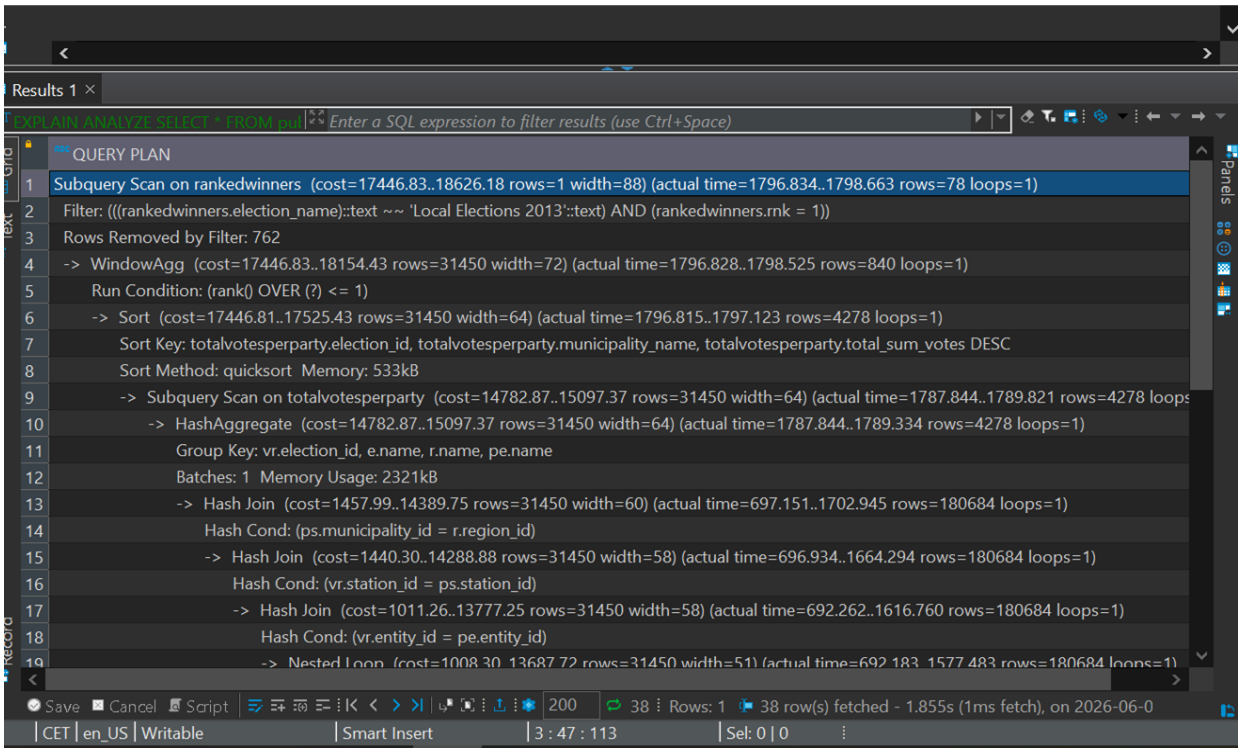
3. Иницијалното време за извршување на погледот изнесува 1855 ms.

```
EXPLAIN ANALYZE

SELECT * FROM public.vw_local_election_winners

WHERE election_name LIKE 'Local Elections 2013';
```

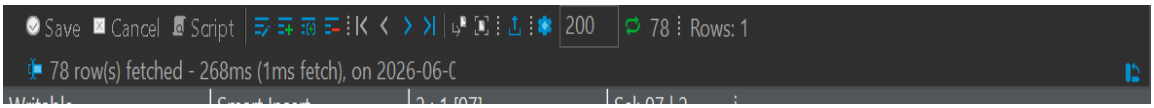


4. Анализата на планот покажува дека базата користи подпрашалници за агрегација (HashAggregate) и сортирање во меморија преку quicksort за аналитичката функција rank(). Бидејќи ова поглед ќе се повикува почесто, времето од 1855 ms се оптимизира со креирање на композитен индекс врз табелата vote_result . Индексот е поставен на колоните (election_id, station_id). Со ова се избегнува бавното скенирање на табелата, се забрзуваат JOIN операциите и директно се оптимизира групирањето на гласовите.

```
CREATE INDEX idx_vote_result_election_station

ON public.vote_result (election_id, station_id);
```

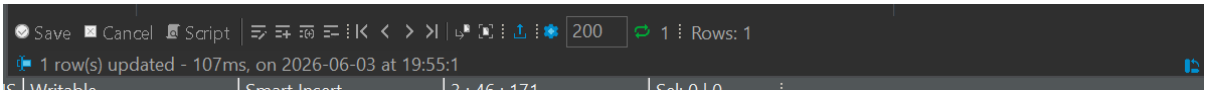


5. Сега иницијално време на извршување изнесува :267ms

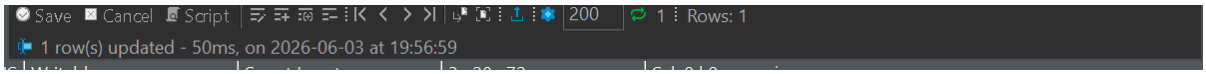


6. Времето изминато во извршување на операциите insert и update врз референтната табела Ballot изнесува:

Input : 107ms



Update : 50ms



9. *vw_parliamentary_municipality_winners*

1. Примарен филтер: election_name (на пр. 'Parliamentary Elections 2014'). Секундарен филтер: municipality_name (доколку сакаме да видиме кој победил во одредена општина на државно ниво).

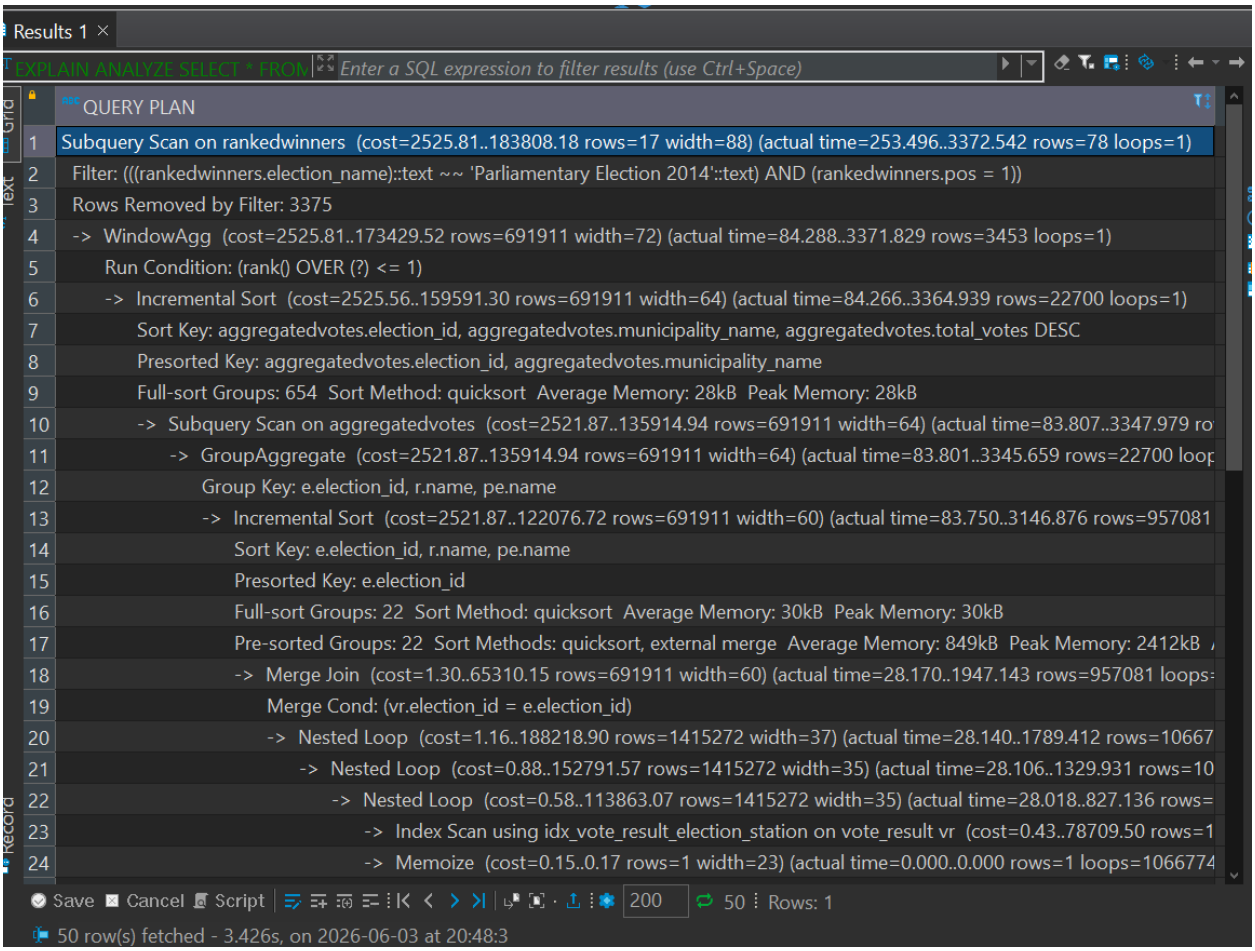
2. Овој поглед служи за детална анализа на изборните резултати од парламентарните избори на ниво на општина. Тој ги агрегира гласовите за секој политички субјект во секоја општина и го идентификува победникот (партијата со најголем број гласови) за таа територијална единица.

3. Иницијално време на извршување 3426ms

```
EXPLAIN ANALYZE

SELECT * FROM public.vw_local_election_winners

WHERE election_name LIKE 'Local Elections 2013';
```

4. Анализата покажува дека базата успешно го користи новокреираниот индекс преку операцијата Index Scan со клучот idx_vote_result_election_station врз табелата vote_result vr. Поради огромниот волумен на податоци (над 691,000 записи), базата

извршува Merge Join во комбинација со оптимизација преку Memoize (ред 24) за да ги избегне повторувачките пресметки. За финално рангирање на доминантната партија се применува Incremental Sort и аналитичката функција во операцијата WindowAgg. Иако вкупното време изнесува 3426 ms поради масивната обработка на гласови на ниво на цела држава, структурата со имплементираниот индекс е максимално оптимизирана за понатамошно користење.



5. Времето изминато во извршување на операциите insert и update врз референтната табела vote_result изнесува:

Insert:

Update:

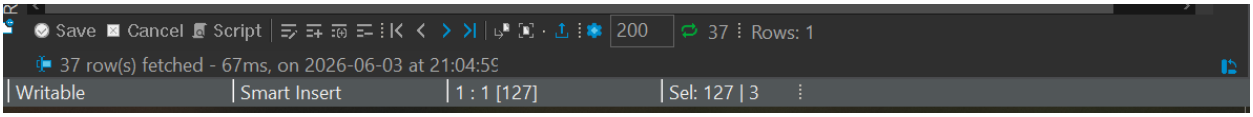
10. vw_presidential_by_municipality

1. Погледот овозможува детален приказ на гласовите освоени од претседателските кандидати, групирани по општина. Тој ги агрегира податоците од табелата Vote Result и ги поврзува со податоците за кандидатите и општините за да генерира извештај за поддршката на кандидатите на локално ниво.

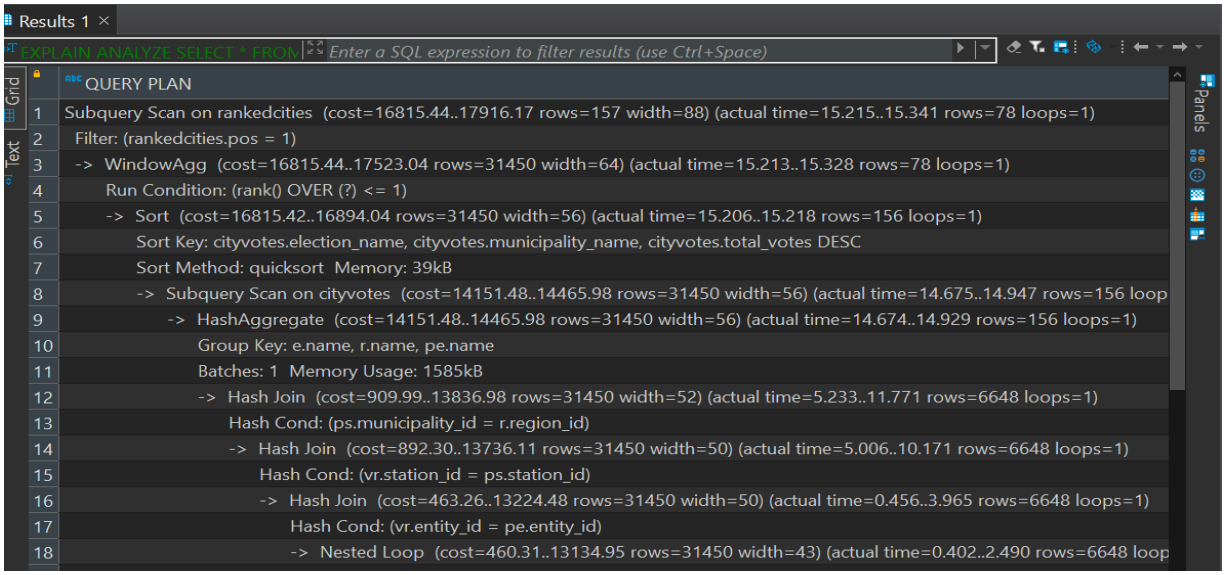
2. Овој извештај ќе се користи за визуелизација на географската распределба на гласовите, каде преку филтрирање по election_name (на пр. 'Presidential Election 2024') и municipality_name корисникот може моментално да ги добие финалните резултати за специфичен регион.

3. Иницијално време на извршување 67ms

```
EXPLAIN ANALYZE

SELECT * FROM public.vw_presidential_by_municipality

WHERE election_name LIKE 'Presidential Election 2024';
```

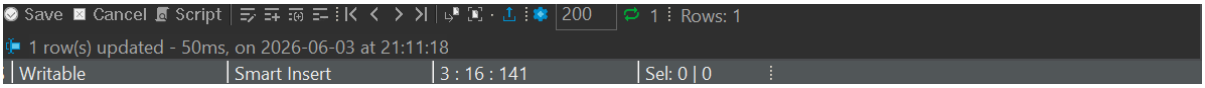


4. Анализата на планот за извршување покажува одлични перформанси, Податоцитеефикасно се поврзуваат преку низа од Hash Join операции и се агрегираат во подпрашалникот cityvotes со помош на HashAggregate. За финално рангирање и извлекување на победниците се користи quicksort во меморија и операцијата WindowAgg. Поради исклучително ниското време на извршување од 67 ms, не се потребни дополнителни оптимизации или нови индекси.

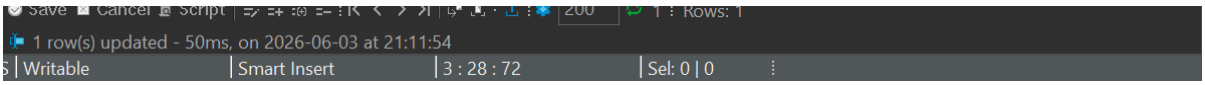


5. Времето изминато во извршување на операциите insert и update врз референтната табела vote_result изнесува:

Insert : 50 ms



Update : 50ms



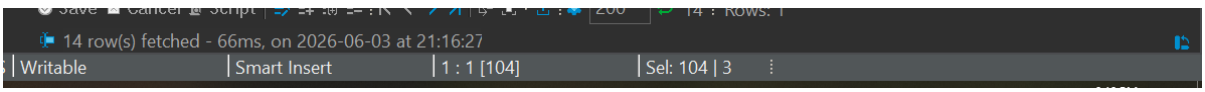
11. *vw_polling_station_voter_count*

1. Примарен филтер за погледот vw_polling_station_voter_count е името на изборното место (station_name) со цел да се провери конкретен капацитет. Дополнително, погледот може да се филтрира и според името на регионот или општината (region_name) за да се анализира густината на гласачкото тело на пошироко локално ниво.

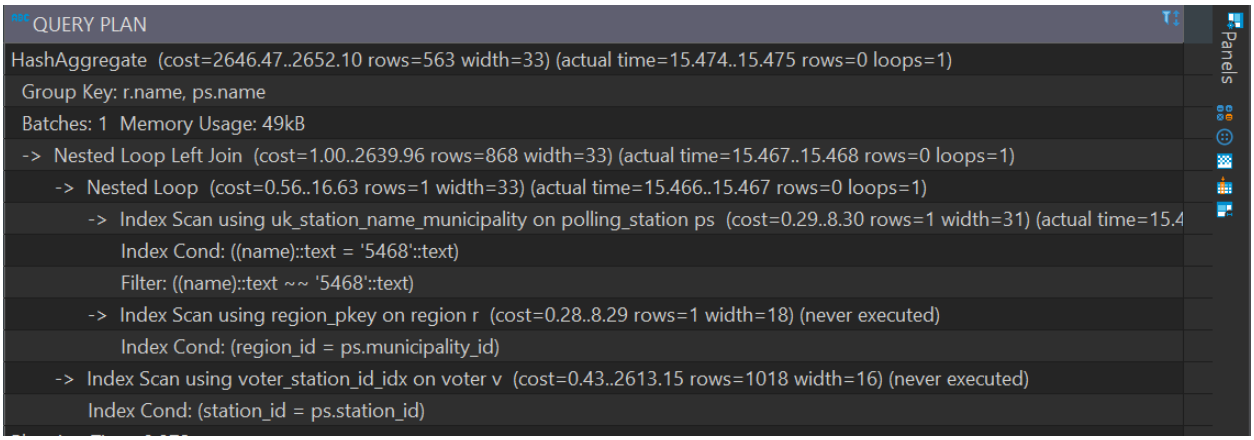
2. Погледот служи за пресметување на вкупниот број на регистрирани гласачи за секое поединечно гласачко место. Наместо да се чуваат статични податоци во табелата polling_station (што би довело до редундантност и потенцијални аномалии), овој поглед врши агрегација во реално време врз основа на поврзаноста помеѓу гласачите и нивните доделени избирачки места.

3. Иницијално време на извршување 66ms

```
EXPLAIN ANALYZE

SELECT * FROM public.vw_polling_station_voter_count

WHERE station_name  LIKE '5468'
```

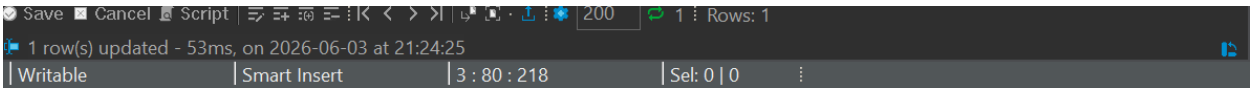


4. Базата работи оптимално бидејќи користи индексни скенирања. Прво се извршува Index Scan со клучот uk_station_name_municipality врз табелата polling_station ps за да се филтрира бараната станица, при што важно е да се напомене дека овој уникатен индекс е автоматски генериран од самиот систем во позадина кога била креирана табелата (UNIQUE CONSTRAINT). Понатаму, преку Nested Loop Left Join се прави поврзување со табелата со гласачи каде исто така е достапен индекс (voter_station_id_idx). Финалното групирање и пребројување на гласачите по изборни места се врши во меморија со помош на ефикасната операција HashAggregate. Поради одличната поставеност на индексите и брзината од 66 ms, не се потребни дополнителни оптимизации.

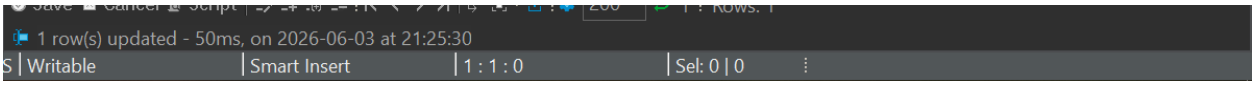


5. Времето изминато во извршување на операциите insert и update врз референтната табела polling_station изнесува:

Insert: 53ms



Update: 50 ms



12. vw_regional_voting_patterns

1. Примарен филтер за vw_regional_voting_patterns е election_id, бидејќи директно го ограничува бројот на записи што се обработуваат од табелата vote_result и овозможува искористување на постоечкиот индекс. Може

дополнително да се филтрира и по region_name за анализа на конкретен регион.

2. View-то служи за анализа на доминантниот политички ентитет по региони за одредени избори. Се користи за утврдување на победникот во секој регион врз основа на вкупниот број освоени гласови и за споредба на регионалните гласачки модели помеѓу различни изборни циклуси.
3. Иницијално време за извршување на погледот е 1798.158ms.

```
EXPLAIN ANALYZE
WITH regionhierarchy AS (
    SELECT ps.station_id,
           r.name AS municipality_name,
           parent.name AS sub_region_name,
           grandparent.name AS country_name
    FROM polling_station ps
    JOIN region r ON ps.municipality_id = r.region_id
    LEFT JOIN region parent ON r.parent_region_id = parent.region_id
    LEFT JOIN region grandparent ON parent.parent_region_id = grandparent.region_id
)
SELECT *
FROM (
    SELECT e.election_id,
           e.name AS election_name,
           pe.name AS candidate_name,
           CASE
               WHEN e.election_type_id = 1 THEN rh.country_name
               ELSE rh.municipality_name
           END AS final_region,
           SUM(vr.votes) AS total_votes_sum
    FROM vote_result vr
    JOIN election e ON vr.election_id = e.election_id
    JOIN political_entity pe ON vr.entity_id = pe.entity_id
    JOIN regionhierarchy rh ON vr.station_id = rh.station_id
    WHERE e.election_id = 8
    GROUP BY e.election_id, e.name, pe.name,
             CASE
                 WHEN e.election_type_id = 1 THEN rh.country_name
                 ELSE rh.municipality_name
             END
) x;
```

Results 1

EXPLAIN ANALYZE WITH regionhierarchy AS (SELECT ps.station_id, r | Enter a SQL expression to filter results (use Ctrl+Space)

Grid | QUERY PLAN

39 | Execution Time: 37.987 ms

4. При филтрирање по election_id е измерено време од 37.987 ms, при што оптимизаторот користи Bitmap Index Scan преку индексот idx_vote_result_election_station, со што се ограничува обработката само на еден изборен циклус. Овој пристап значително го намалува бројот на обработени редови и го подобрува извршувањето.
   Нема потреба од дополнителна оптимизација. View-то има добри перформанси кога се користи election_id, бидејќи тогаш се активира индексен пристап и извршувањето останува под 40 ms.

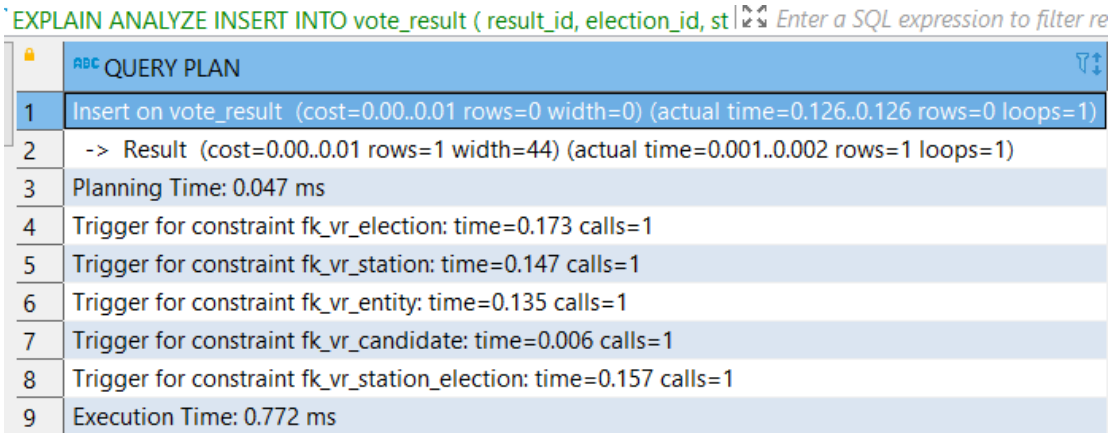
```
EXPLAIN ANALYZE WITH regionhierarchy AS ( SELECT ps.station_id, r
```

	QUERY PLAN
1	HashAggregate (cost=14075.18..14281.58 rows=16512 width=272) (actual time=37.660..37.796 rows=391 loops=1)
2	Group Key: pe.name, CASE WHEN (e.election_type_id = 1) THEN grandparent.name ELSE r.name END
3	Batches: 1 Memory Usage: 465kB
4	-> Hash Left Join (cost=729.46..13951.34 rows=16512 width=268) (actual time=5.661..31.100 rows=16621 loops=1)
5	Hash Cond: (parent.parent_region_id = grandparent.region_id)
6	-> Hash Left Join (cost=711.77..13848.86 rows=16512 width=76) (actual time=5.464..26.947 rows=16621 loops=1)
7	Hash Cond: (r.parent_region_id = parent.region_id)
8	-> Hash Join (cost=694.08..13787.66 rows=16512 width=76) (actual time=5.272..23.139 rows=16621 loops=1)
9	Hash Cond: (ps.municipality_id = r.region_id)
10	-> Hash Join (cost=676.39..13726.30 rows=16512 width=66) (actual time=5.036..18.503 rows=16621 loops=1)
11	Hash Cond: (vr.station_id = ps.station_id)
12	-> Hash Join (cost=247.35..13253.90 rows=16512 width=66) (actual time=0.678..9.973 rows=16621 loops=1)
13	Hash Cond: (vr.entity_id = pe.entity_id)
14	-> Nested Loop (cost=244.40..13205.49 rows=16512 width=59) (actual time=0.618..6.062 rows=16621 loops=1)
15	-> Seq Scan on election e (cost=0.00..1.56 rows=1 width=39) (actual time=0.011..0.016 rows=1 loops=1)
16	Filter: (election_id = 8)
17	Rows Removed by Filter: 44
18	-> Bitmap Heap Scan on vote_result vr (cost=244.40..13038.81 rows=16512 width=28) (actual time=0.603..3.0
19	Recheck Cond: (election_id = 8)
20	Heap Blocks: exact=142
21	-> Bitmap Index Scan on idx_vote_result_election_station (cost=0.00..240.27 rows=16512 width=0) (actual
22	Index Cond: (election_id = 8)
23	-> Hash (cost=1.87..1.87 rows=87 width=23) (actual time=0.039..0.040 rows=87 loops=1)
24	Buckets: 1024 Batches: 1 Memory Usage: 14kB
25	-> Seq Scan on political_entity pe (cost=0.00..1.87 rows=87 width=23) (actual time=0.010..0.022 rows=87 loo
26	-> Hash (cost=281.24..281.24 rows=11824 width=16) (actual time=4.327..4.327 rows=11825 loops=1)
27	Buckets: 16384 Batches: 1 Memory Usage: 683kB
28	-> Seq Scan on polling_station ps (cost=0.00..281.24 rows=11824 width=16) (actual time=0.011..2.263 rows=1182
29	-> Hash (cost=10.64..10.64 rows=564 width=26) (actual time=0.228..0.228 rows=564 loops=1)
30	Buckets: 1024 Batches: 1 Memory Usage: 41kB
31	-> Seq Scan on region r (cost=0.00..10.64 rows=564 width=26) (actual time=0.022..0.112 rows=564 loops=1)
32	-> Hash (cost=10.64..10.64 rows=564 width=16) (actual time=0.185..0.185 rows=564 loops=1)
33	Buckets: 1024 Batches: 1 Memory Usage: 33kB
34	-> Seq Scan on region parent (cost=0.00..10.64 rows=564 width=16) (actual time=0.011..0.091 rows=564 loops=1)
35	-> Hash (cost=10.64..10.64 rows=564 width=18) (actual time=0.189..0.189 rows=564 loops=1)

Времето изминато во извршување на операциите insert и update изнесува:

```
EXPLAIN ANALYZE UPDATE vote_result SET votes = votes + 1 WHERE
```

	QUERY PLAN
1	Update on vote_result (cost=0.43..8.45 rows=0 width=0) (actual time=0.181..0.182 rows=0 loops=1)
2	-> Index Scan using idx_vote_result_election_station on vote_result (cost=0.43..8.45 rows=1 width=10) (actual time=0.050..0.052 rows=1 loops=1)
3	Index Cond: ((election_id = 8) AND (station_id = 6114))
4	Filter: (entity_id = 1004)
5	Rows Removed by Filter: 5
6	Planning Time: 0.140 ms
7	Execution Time: 0.221 ms



5. Нема потреба од дополнителна оптимизација.
6. Време на извршување останува исто.