

Напредни бази на податоци

Фаза 3: Оптимизација на прашалници и погледи

Проект: Платформа за евиденција на избори EDB



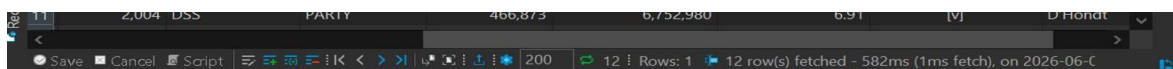
1. vw_election_results

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.

The screenshot shows a query plan for an EXPLAIN ANALYZE SELECT statement. The plan consists of the following steps:

- 1. Nested Loop (cost=38626.33..38666.44 rows=45 width=769) (actual time=179.197..185.339 rows=12 loops=1)
 - 2. Seq Scan on political_entity pe (cost=0.00..2.09 rows=1 width=31) (actual time=0.022..0.032 rows=1 loops=1)
 - 3. Filter: (entity_id = 2004)
 - 4. Rows Removed by Filter: 86
 - 5. Hash Join (cost=38626.33..38662.55 rows=45 width=713) (actual time=179.158..185.268 rows=12 loops=1)
 - 6. Hash Cond: (e.winner_method_id = wm.method_id)
 - 7. Hash Join (cost=38625.04..38661.12 rows=45 width=205) (actual time=179.130..185.236 rows=12 loops=1)
 - 8. Hash Cond: (e.election_type_id = et.election_type_id)
 - 9. Hash Join (cost=38623.84..38659.74 rows=45 width=75) (actual time=179.110..185.211 rows=12 loops=1)
 - 10. Hash Cond: (vote_result.election_id = e.election_id)
 - 11. Merge Join (cost=38621.83..38657.60 rows=45 width=40) (actual time=179.056..185.148 rows=12 loops=1)
 - 12. Merge Cond: (vote_result.election_id = vote_result_1.election_id)
 - 13. Finalize GroupAggregate (cost=18813.01..18830.11 rows=45 width=24) (actual time=58.627..58.769 rows=12 loops=1)
 - 14. Group Key: vote_result.election_id
 - 15. Gather Merge (cost=18813.01..18828.99 rows=135 width=24) (actual time=58.617..58.748 rows=48 loops=1)
 - 16. Workers Planned: 3
 - 17. Workers Launched: 3
 - 18. Sort (cost=17812.97..17813.09 rows=45 width=24) (actual time=51.974..51.977 rows=12 loops=1)
 - 19. Sort Key: vote_result.election_id
 - 20. Sort Method: quicksort Memory: 25kB
 - 21. Worker 0: Sort Method: quicksort Memory: 25kB
 - 22. Worker 1: Sort Method: quicksort Memory: 25kB
 - 23. Worker 2: Sort Method: quicksort Memory: 25kB

The status bar at the bottom indicates: 200 rows, 61 rows fetched, 275ms execution time on 2026-06-03 at 18:41:2.

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

Insert : 219 ms

The screenshot shows a status bar with the following information: 1 row(s) updated - 52ms, on 2026-06-03 at 18:58:29.

Update: 75ms

The screenshot shows a status bar with the following information: 1 row(s) updated - 75ms, on 2026-06-03 at 18:59:34.

2. party_performance_over_time

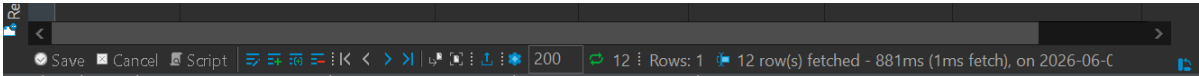
1. Примарен филтер за погледот vw_party_performance_over_time е идентификаторот на политичкиот субјект (entity_id), бидејќи корисниците најчесто го анализираат историскиот успех и трендот на една конкретна партија. Дополнително, погледот може да се филтрира и според датумот на изборите (election_date) 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 графיקони кои го прикажуваат историскиот перформанс на политичките субјекти.

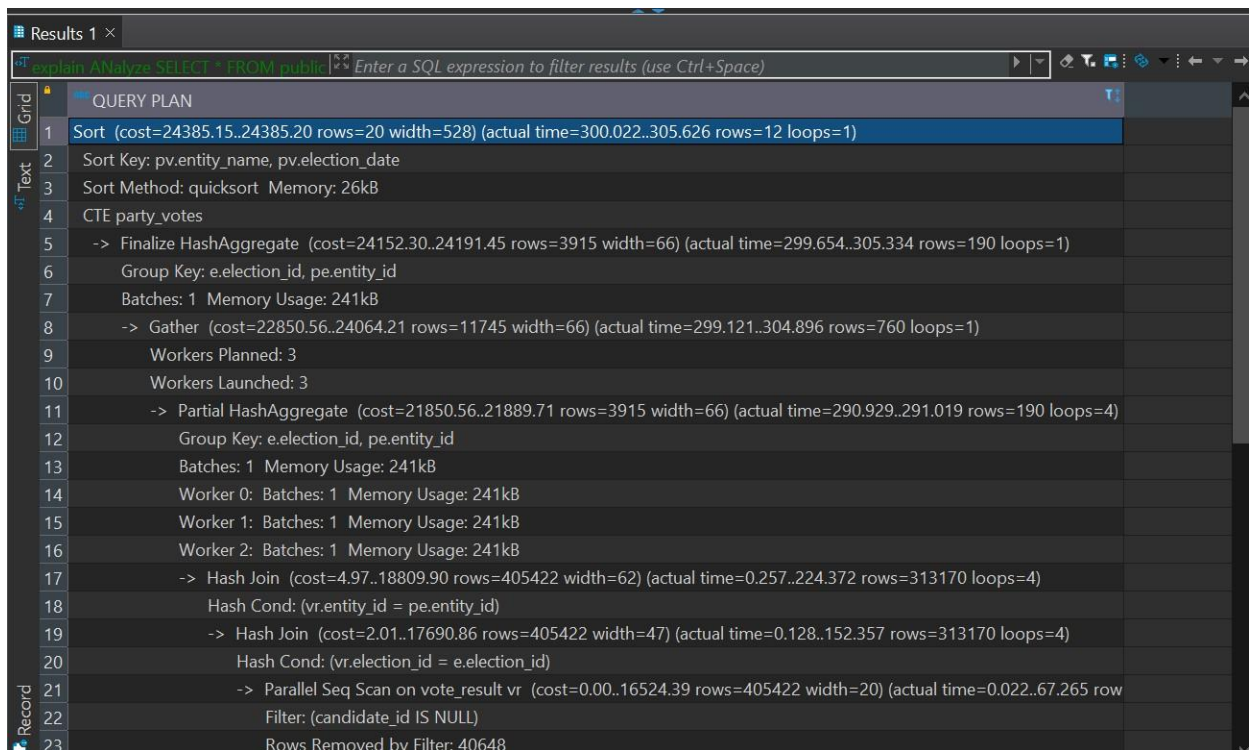
3. Иницијално време за извршување е 880 ms

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.

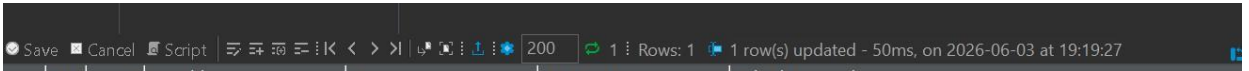


5. Времето изминато во извршување на операциите insert и update

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.

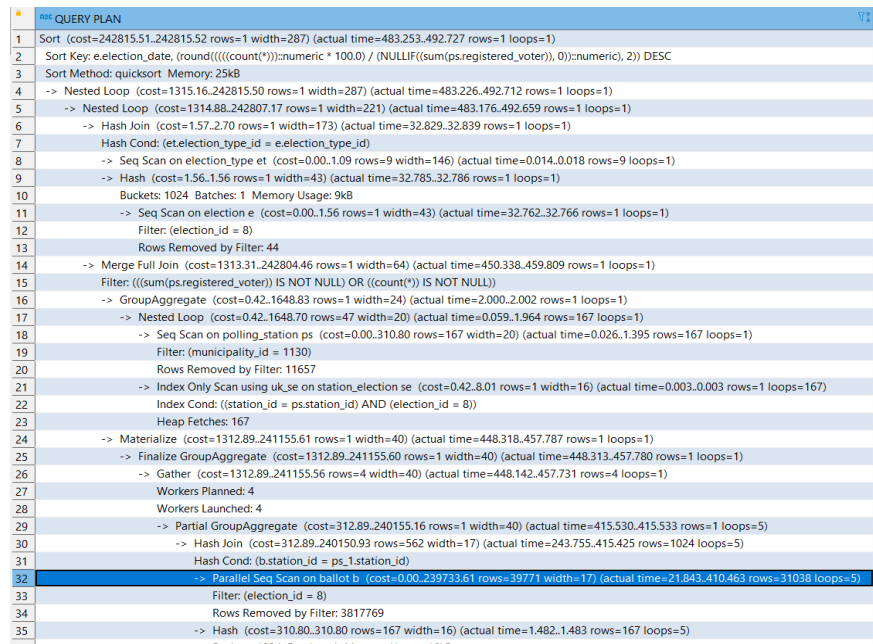
EXPLAIN ANALYZE SELECT * FROM vw_voter_turnout WHERE election_id = 8



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:

| ABC 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 |

| ABC 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);
```

| ABC QUERY PLAN | |
|----------------|--|
| 1 | Rows Removed by Filter: 11657 |
| 2 | -> 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=1) |
| 3 | Index Cond: (region_id = 1130) |
| 4 | Planning Time: 1.119 ms |
| 5 | JIT: |
| 6 | Functions: 117 |
| 7 | Options: Inlining false, Optimization false, Expressions true, Deforming true |
| 8 | Timing: Generation 8.704 ms (Deform 4.032 ms), Inlining 0.000 ms, Optimization 3.668 ms, Emission 84.196 ms, Total 96.569 ms |
| 9 | Execution Time: 101.735 ms |

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

| ABC QUERY PLAN | |
|----------------|---|
| 1 | 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 |

| ABC 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_performance
```

| ABC 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_performe | Enter a SQL expression to filter results (use Ctrl+Space)

| ABC 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 извршување 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)

| ABC 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 filter results (use Ctrl+Space)

| ABC 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 |

- Нема потреба од дополнителна оптимизација.
- 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)

abc 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.

EXPLAIN ANALYZE SELECT ps.station_id, COUNT(b.ballot_id) AS total FROM polling_station ps JOIN ballot b ON ps.station_id = b.station_id

| Grid | ABC QUERY PLAN |
|------|--|
| 1 | HashAggregate (cost=244153.15..244271.39 rows=11824 width=16) (actual time=113.694..115.677 rows=11824 loops=1) |
| 2 | Group Key: ps.station_id |
| 3 | Batches: 1 Memory Usage: 1169kB |
| 4 | -> Hash Right Join (cost=3201.36..243378.17 rows=154996 width=16) (actual time=20.123..85.787 rows=159753 loops=1) |
| 5 | Hash Cond: (b.station_id = ps.station_id) |
| 6 | -> Gather (cost=2772.32..242542.12 rows=154996 width=16) (actual time=6.375..40.343 rows=151253 loops=1) |
| 7 | Workers Planned: 4 |
| 8 | Workers Launched: 4 |
| 9 | -> Parallel Bitmap Heap Scan on ballot b (cost=1772.32..226042.52 rows=38749 width=16) (actual time=6.467..12.908 rows=30251 loops=5) |
| 10 | Recheck Cond: (election_id = 8) |
| 11 | Filter: is_valid |
| 12 | Rows Removed by Filter: 788 |
| 13 | Heap Blocks: exact=507 |
| 14 | -> Bitmap Index Scan on idx_ballot_election_id (cost=0.00..1733.57 rows=159084 width=0) (actual time=5.683..5.683 rows=155191 loops=1) |
| 15 | Index Cond: (election_id = 8) |
| 16 | -> Hash (cost=281.24..281.24 rows=11824 width=8) (actual time=13.703..13.704 rows=11824 loops=1) |
| 17 | Buckets: 16384 Batches: 1 Memory Usage: 590kB |
| 18 | -> Seq Scan on polling_station ps (cost=0.00..281.24 rows=11824 width=8) (actual time=9.792..11.787 rows=11824 loops=1) |
| 19 | Planning Time: 0.431 ms |
| 20 | JIT: |
| 21 | Functions: 42 |
| 22 | Options: Inlining false, Optimization false, Expressions true, Deforming true |
| 23 | Timing: Generation 3.120 ms (Deform 1.454 ms), Inlining 0.000 ms, Optimization 2.124 ms, Emission 33.663 ms, Total 38.907 ms |
| 24 | Execution Time: 117.591 ms |

Времето изминато во извршување на операциите insert и update изнесува:

EXPLAIN ANALYZE INSERT INTO ballot (election_id, station_id, entity)

| Grid | ABC QUERY PLAN |
|------|---|
| 1 | Insert on ballot (cost=0.00..0.02 rows=0 width=0) (actual time=0.706..0.706 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: 0.053 ms |
| 4 | Trigger for constraint fk_ballot_election: time=0.122 calls=1 |
| 5 | Trigger for constraint fk_ballot_station: time=0.106 calls=1 |
| 6 | Trigger for constraint fk_ballot_entity: time=0.099 calls=1 |
| 7 | Trigger for constraint fk_ballot_candidate: time=0.006 calls=1 |
| 8 | Trigger trg_after_insert_ballot_live_count: time=0.290 calls=1 |
| 9 | Trigger trg_validate_ballot: time=0.606 calls=1 |
| 10 | Execution Time: 1.369 ms |

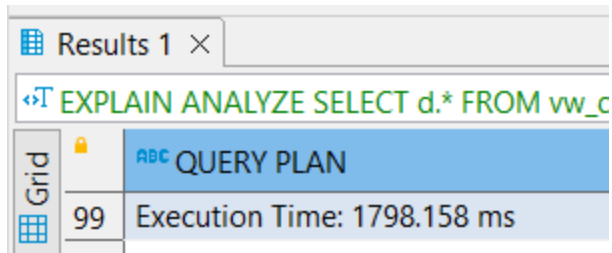
EXPLAIN ANALYZE UPDATE ballot SET is_valid = false WHERE ballot_id = 100000

| Grid | ABC QUERY PLAN |
|------|--|
| 1 | Update on ballot (cost=0.44..8.46 rows=0 width=0) (actual time=0.024..0.025 rows=0 loops=1) |
| 2 | -> Index Scan using ballot_pkey on ballot (cost=0.44..8.46 rows=1 width=7) (actual time=0.024..0.025 rows=1 loops=1) |
| 3 | Index Cond: (ballot_id = 100000) |
| 4 | Planning Time: 0.145 ms |
| 5 | Execution Time: 0.060 ms |

- Нема потреба од дополнителна оптимизација.
- Време на извршување останува исто.

C. 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)*

| Step | Operation | Cost | Time | Rows | Width | Actual Time | Actual Rows | Loops |
|------|--|------------------------|------|--------|-------|--------------------|-------------|-------|
| 1 | Hash Join | 1560212.77..1560496.42 | | 6023 | 168 | 1789.704..1789.942 | 286 | 1 |
| 2 | Hash Cond: (ranked.election_id = e.election_id) | | | | | | | |
| 3 | -> Sort | 1560210.93..1560347.81 | | 54751 | 168 | 629.775..629.950 | 286 | 1 |
| 4 | Sort Key: ranked.election_id, ranked.district_id, (count(*) FILTER (WHERE (ranked.rn <= ranked.seats_available))) DESC | | | | | | | |
| 5 | Sort Method: quicksort | | | | | | | |
| 6 | Memory: 66kB | | | | | | | |
| 7 | CTE district_votes | | | | | | | |
| 8 | -> HashAggregate | 23907.55..24075.36 | | 16781 | 44 | 601.620..602.838 | 3391 | 1 |
| 9 | Group Key: ed.district_id, vr.entity_id | | | | | | | |
| 10 | Batches: 1 | | | | | | | |
| 11 | Memory Usage: 1297kB | | | | | | | |
| 12 | -> Gather | 2484.39..23781.70 | | 16781 | 40 | 85.710..552.937 | 277081 | 1 |
| 13 | Workers Planned: 3 | | | | | | | |
| 14 | Workers Launched: 3 | | | | | | | |
| 15 | -> Hash Join | 1484.39..21103.60 | | 5413 | 40 | 434.864..514.695 | 69270 | 4 |
| 16 | Hash Cond: ((vr.election_id = e_2.election_id) AND (vr.station_id = ps.station_id)) | | | | | | | |
| 17 | -> Parallel Seq Scan on vote_result vr | 0.00..16524.39 | | 405422 | 28 | 376.311..416.897 | 313171 | 4 |
| 18 | Filter: (candidate_id IS NULL) | | | | | | | |
| 19 | Rows Removed by Filter: 40648 | | | | | | | |
| 20 | -> Hash | 1377.78..1377.78 | | 7107 | 44 | 49.025..49.030 | 52100 | 4 |
| 21 | Buckets: 65536 (originally 8192) Batches: 1 (originally 1) Memory Usage: 4384kB | | | | | | | |
| 22 | -> Hash Join | 981.13..1377.78 | | 7107 | 44 | 22.134..31.332 | 52100 | 4 |
| 23 | Hash Cond: (ps.municipality_id = r2.region_id) | | | | | | | |
| 24 | -> Seq Scan on polling_station ps | 0.00..281.24 | | 11824 | 16 | 0.045..1.896 | 11825 | 4 |
| 25 | -> Hash | 976.90..976.90 | | 339 | 44 | 22.048..22.052 | 1744 | 4 |
| 26 | Buckets: 2048 (originally 1024) Batches: 1 (originally 1) Memory Usage: 145kB | | | | | | | |
| 27 | -> Nested Loop | 6.46..976.90 | | 339 | 44 | 0.293..21.367 | 1744 | 4 |
| 28 | -> Hash Join | 1.62..20.74 | | 105 | 36 | 0.129..0.605 | 684 | 4 |
| 29 | Hash Cond: (ed.election_id = e_2.election_id) | | | | | | | |
| 30 | -> Seq Scan on electoral_district ed | 0.00..16.41 | | 941 | 28 | 0.034..0.244 | 941 | 4 |
| 31 | -> Hash | 1.56..1.56 | | 5 | 8 | 0.071..0.071 | 25 | 4 |
| 32 | Buckets: 1024 Batches: 1 Memory Usage: 9kB | | | | | | | |
| 33 | -> Seq Scan on election e_2 | 0.00..1.56 | | 5 | 8 | 0.052..0.058 | 25 | 4 |
| 34 | Filter: (winner_method_id = 8) | | | | | | | |
| 35 | Rows Removed by Filter: 20 | | | | | | | |
| 36 | -> Bitmap Heap Scan on region r2 | 4.83..9.00 | | 11 | 16 | 0.027..0.027 | 3 | 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 results (use Ctrl+Space)*

| Step | Operation | Cost | Time | Rows | Width | Actual Time | Actual Rows | Loops |
|------|---|------------|------|------|-------|--------------|-------------|-------|
| 1 | Insert on vote_result | 0.00..0.01 | | 0 | 0 | 1.544..1.545 | 0 | 1 |
| 2 | -> Result | 0.00..0.01 | | 1 | 44 | 0.001..0.002 | 1 | 1 |
| 3 | Planning Time: 0.047 ms | | | | | | | |
| 4 | Trigger for constraint fk_vr_election: time=0.421 calls=1 | | | | | | | |
| 5 | Trigger for constraint fk_vr_station: time=0.238 calls=1 | | | | | | | |
| 6 | Trigger for constraint fk_vr_entity: time=0.313 calls=1 | | | | | | | |
| 7 | Trigger for constraint fk_vr_candidate: time=0.085 calls=1 | | | | | | | |
| 8 | Trigger for constraint fk_vr_station_election: time=0.529 calls=1 | | | | | | | |
| 9 | Execution Time: 3.156 ms | | | | | | | |

EXPLAIN ANALYZE UPDATE vote_result SET votes = votes + 50 WHEI *Enter a SQL expression to filter results (use Ctrl+Space)*

| Step | Operation | Cost | Time | Rows | Width | Actual Time | Actual Rows | Loops |
|------|---|---------------|------|------|-------|--------------|-------------|-------|
| 1 | Update on vote_result | 11.67..819.66 | | 0 | 0 | 0.152..0.154 | 0 | 1 |
| 2 | -> Bitmap Heap Scan on vote_result | 11.67..819.66 | | 226 | 10 | 0.082..0.082 | 226 | 1 |
| 3 | Recheck Cond: ((election_id = 8) AND (entity_id = 1004) AND (candidate_id IS NULL)) | | | | | | | |
| 4 | Heap Blocks: exact=1 | | | | | | | |
| 5 | -> Bitmap Index Scan on uk_vote_result | 0.00..11.62 | | 226 | 0 | 0.000..0.000 | 226 | 1 |
| 6 | Index Cond: ((election_id = 8) AND (entity_id = 1004) AND (candidate_id IS NULL)) | | | | | | | |
| 7 | Planning Time: 0.147 ms | | | | | | | |
| 8 | Execution Time: 0.292 ms | | | | | | | |

- Нема потреба од дополнителна оптимизација.
- Време на извржување останува исто

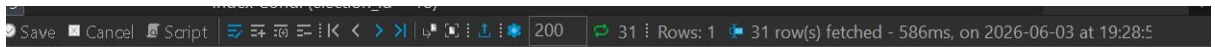
7. *vw_invalid_ballot_analysis*

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).

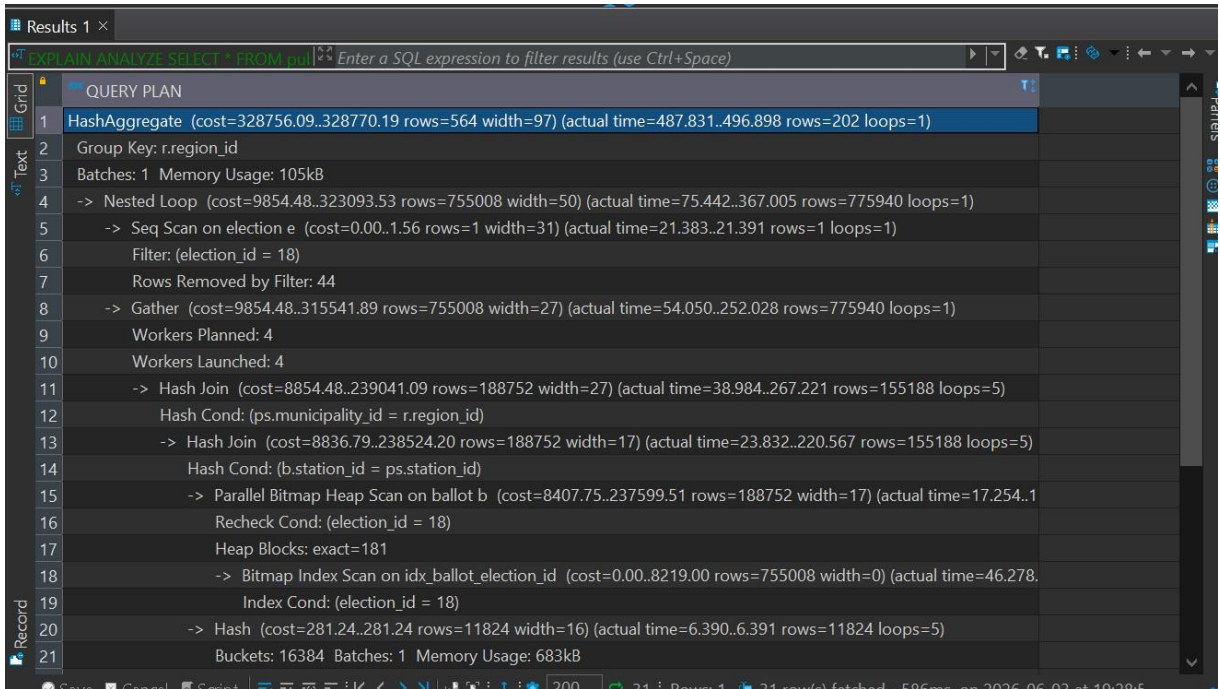
EXPLAIN ANALYZE

```
SELECT * FROM public.vw_invalid_ballot_analysis
```

```
WHERE election_id = 18;
```

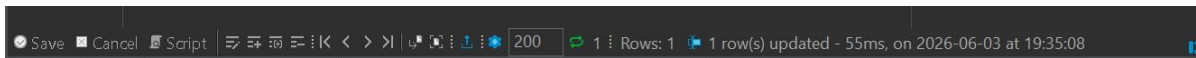


4. Анализата на планот за извршување покажува дека базата успешно го користи постоечкиот индекс врз табелата `ballot` преку операцијата `Bitmap Index Scan (idx_ballot_election_id)`, со што директно се таргетираат бараните записи. Бидејќи индексот е веќе имплементиран и перформансите се оптимизирани,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

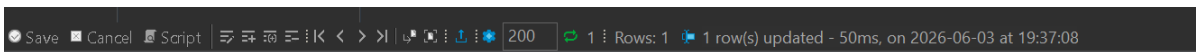


5. Времето изминато во извршување на операциите insert и update врз референтната табела Ballot изнесува:

Insert : 55ms



Update:



8. *vw_local_election_winners*

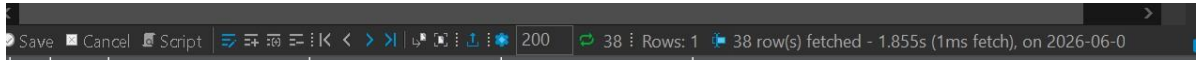
1. Примарен филтер за погледот *vw_local_election_winners* (на пр. 'Local Elections 2013') е името на изборите (*election_name*), бидејќи анализата на победниците секогаш се врши за одреден изборен циклус .

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.

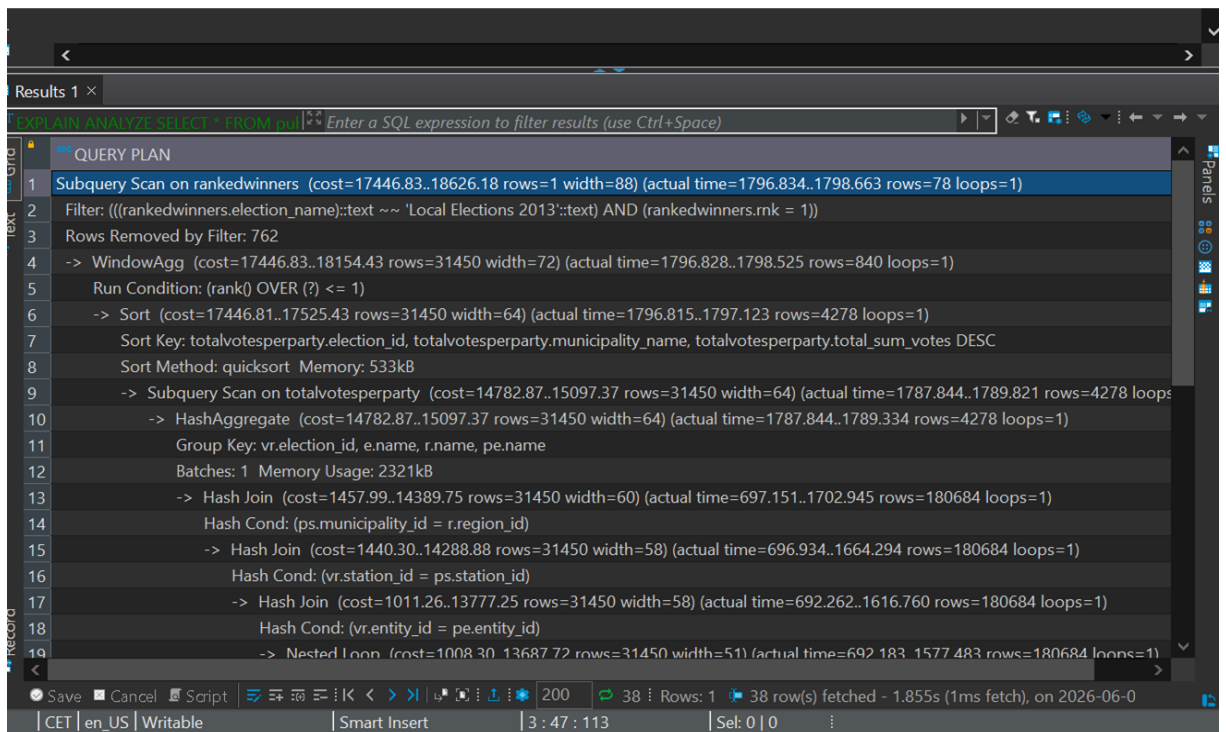
EXPLAIN ANALYZE

```
SELECT * FROM public.vw_local_election_winners  
  
WHERE election_name LIKE 'Local Elections 2013';
```



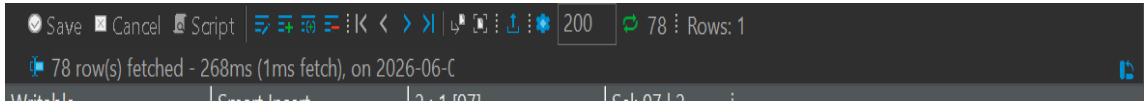
4. Анализата на планот покажува дека базата користи подпрашалници за агрегација (HashAggregate) и сортирање во меморија преку quicksort за аналитичката функција rank(). Бидејќи ова поглед ќе се повикува почесто, времето од 1855 ms се оптимизира со креирање на композитен индекс врз табелата vote_result . 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);
```



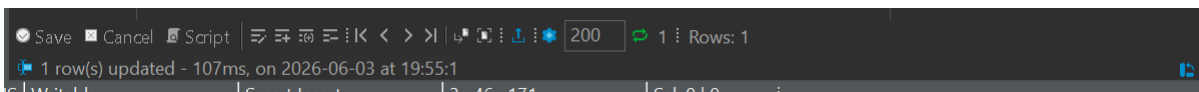
```
Results 1 x  
EXPLAIN ANALYZE SELECT * FROM public.vw_local_election_winners WHERE election_name LIKE 'Local Elections 2013';  
Enter a SQL expression to filter results (use Ctrl+Space)  
QUERY PLAN  
1 Subquery Scan on rankedwinners (cost=17446.83..18626.18 rows=1 width=88) (actual time=1796.834..1798.663 rows=78 loops=1)  
2 Filter: (((rankedwinners.election_name)::text ~~ 'Local Elections 2013'::text) AND (rankedwinners.rnk = 1))  
3 Rows Removed by Filter: 762  
4 -> WindowAgg (cost=17446.83..18154.43 rows=31450 width=72) (actual time=1796.828..1798.525 rows=840 loops=1)  
5 Run Condition: (rank() OVER (?) <= 1)  
6 -> Sort (cost=17446.81..17525.43 rows=31450 width=64) (actual time=1796.815..1797.123 rows=4278 loops=1)  
7 Sort Key: totalvotesperparty.election_id, totalvotesperparty.municipality_name, totalvotesperparty.total_sum_votes DESC  
8 Sort Method: quicksort Memory: 533kB  
9 -> Subquery Scan on totalvotesperparty (cost=14782.87..15097.37 rows=31450 width=64) (actual time=1787.844..1789.821 rows=4278 loops=1)  
10 -> HashAggregate (cost=14782.87..15097.37 rows=31450 width=64) (actual time=1787.844..1789.334 rows=4278 loops=1)  
11 Group Key: vr.election_id, e.name, r.name, pe.name  
12 Batches: 1 Memory Usage: 2321kB  
13 -> Hash Join (cost=1457.99..14389.75 rows=31450 width=60) (actual time=697.151..1702.945 rows=180684 loops=1)  
14 Hash Cond: (ps.municipality_id = r.region_id)  
15 -> Hash Join (cost=1440.30..14288.88 rows=31450 width=58) (actual time=696.934..1664.294 rows=180684 loops=1)  
16 Hash Cond: (vr.station_id = ps.station_id)  
17 -> Hash Join (cost=1011.26..13777.25 rows=31450 width=58) (actual time=692.262..1616.760 rows=180684 loops=1)  
18 Hash Cond: (vr.entity_id = pe.entity_id)  
19 -> Nested Loop (cost=1008.30..13687.72 rows=31450 width=51) (actual time=692.183..1577.483 rows=180684 loops=1)
```

5. Сега иницијално време на извршување изнесува :267ms

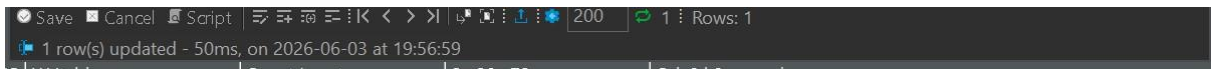


6. Времето изменато во извршување на операциите insert и update врз референтната табела Ballot изнесува:

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.

The screenshot displays a query plan for an SQL query. The plan starts with a Subquery Scan on 'rankedwinners' (cost=2525.81..183808.18 rows=17 width=88). This is followed by a filter: '(((rankedwinners.election_name)::text ~~ 'Parliamentary Election 2014)::text) AND (rankedwinners.pos = 1)'. The plan then shows a WindowAgg operation (cost=2525.81..173429.52 rows=691911 width=72) with a run condition 'rank() OVER (?) <= 1'. This is followed by an Incremental Sort (cost=2525.56..159591.30 rows=691911 width=64) with sort keys 'aggregatedvotes.election_id, aggregatedvotes.municipality_name, aggregatedvotes.total_votes DESC'. The plan continues with a Subquery Scan on 'aggregatedvotes' (cost=2521.87..135914.94 rows=691911 width=64), a GroupAggregate (cost=2521.87..135914.94 rows=691911 width=64), another Incremental Sort (cost=2521.87..122076.72 rows=691911 width=60), and a Merge Join (cost=1.30..65310.15 rows=691911 width=60) with a merge condition 'vr.election_id = e.election_id'. The final step is a Nested Loop (cost=1.16..188218.90 rows=1415272 width=37), which contains another Nested Loop (cost=0.88..152791.57 rows=1415272 width=35), which in turn contains a third Nested Loop (cost=0.58..113863.07 rows=1415272 width=35), and finally an Index Scan using 'idx_vote_result_election_station' on 'vote_result vr' (cost=0.43..78709.50 rows=1) and a Memoize operation (cost=0.15..0.17 rows=1 width=23). The bottom of the screenshot shows '50 row(s) fetched - 3.426s, on 2026-06-03 at 20:48:3'.

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

Insert:

The screenshot shows a database console output for an insert operation: '1 row(s) updated - 50ms, on 2026-06-03 at 20:56:37'. Below this, there are status indicators: 'Writable', 'Smart Insert', '3 : 46 [171]', and 'Sel: 171 | 3'.

Update:

The screenshot shows a database console output for an update operation: '1 row(s) updated - 49ms, on 2026-06-03 at 20:57:12'. Below this, there are status indicators: 'Writable', 'Smart Insert', '3 : 46 [171]', and 'Sel: 171 | 3'.

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.

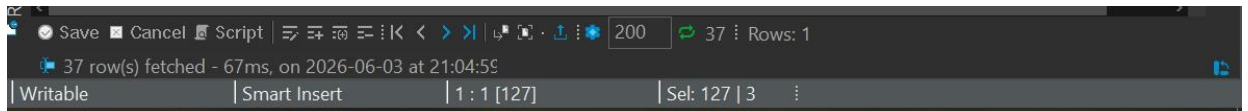
2. Овој извештај ќе се користи за визуелизација на географската распределба на гласовите, каде преку филтрирање по election_name (на пр. 'Presidential Election 2024') и municipality_name корисникот може моментално да ги добие финалните резултати за специфичен регион.

3. Иницијално време на извршување 67ms

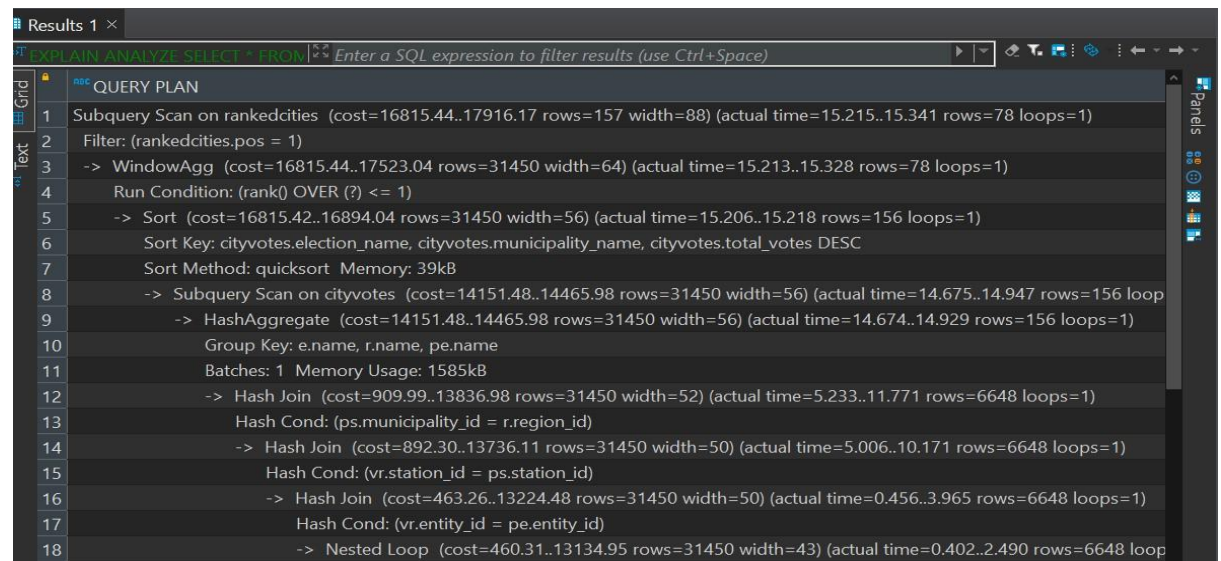
EXPLAIN ANALYZE

```
SELECT * FROM public.vw_presidential_by_municipality
```

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

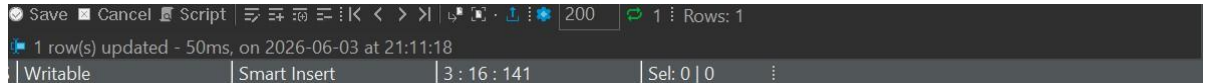


4. Анализата на планот за извршување покажува одлични перформанси, Податоците ефикасно се поврзуваат преку низа од Hash Join операции и се агрегираат во подпрашалникот cityvotes со помош на HashAggregate. За финално рангирање и извлекување на победниците се користи quicksort во меморија и операцијата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

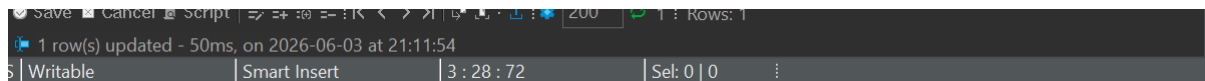


5. Времето изминато во извршување на операциите insert и update врз референтната табела vote_result изнесува:

Insert : 50 ms



Update : 50ms



11. vw_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 x

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

Grid

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

Enter a SQL expression to filter results (use Ctrl+Space)

| ABC 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 loops=1) |
| 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=11825 loops=1) |
| 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) |
| 36 | Buckets: 1024 Batches: 1 Memory Usage: 38kB |

Времето изминато во извршување на операциите insert и update изнесува:

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

| ABC 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 |

```
EXPLAIN ANALYZE INSERT INTO vote_result ( result_id, election_id, st
```

| ABC QUERY PLAN | |
|----------------|--|
| 1 | Insert on vote_result (cost=0.00..0.01 rows=0 width=0) (actual time=0.126..0.126 rows=0 loops=1) |
| 2 | -> Result (cost=0.00..0.01 rows=1 width=44) (actual time=0.001..0.002 rows=1 loops=1) |
| 3 | Planning Time: 0.047 ms |
| 4 | Trigger for constraint fk_vr_election: time=0.173 calls=1 |
| 5 | Trigger for constraint fk_vr_station: time=0.147 calls=1 |
| 6 | Trigger for constraint fk_vr_entity: time=0.135 calls=1 |
| 7 | Trigger for constraint fk_vr_candidate: time=0.006 calls=1 |
| 8 | Trigger for constraint fk_vr_station_election: time=0.157 calls=1 |
| 9 | Execution Time: 0.772 ms |

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

